MASTER'S THESIS

Welke invloed heeft de Online Leeromgeving Grande Omega op Programmeerprestaties en Computational Thinking Skills?

Van Brussel, Anne

Award date:
2020

THESIS

Onderwijswetenschappen

Open Universiteit

Welke invloed heeft de Online Leeromgeving Grande Omega op Programmeerprestaties en Computational Thinking Skills?

What Influence does the Online Learning Program Grande Omega have on Programming Performance and Computational Thinking Skills?

Anne van Brussel

Master Onderwijswetenschappen
Open Universiteit

Datum: 5 mei 2019
Begeleiding: Prof. Dr. Marcus (M.M.) Specht

Inhoudsopgave

Welk invloed heeft de Online Leeromgeving Grande Omega op Programmeerprestaties en Computational Thinking Skills?

Anne van Brussel

Samenvatting

De informaticaopleiding van Hogeschool Rotterdam zoekt naar innoverende manieren om groepen beginnende informatici te leren programmeren. Enkele docenten van de informatica-opleiding van Hogeschool Rotterdam hebben het online leerprogramma Grande Omega ontwikkeld, waarmee alle studenten vanaf het eerste jaar van hun opleiding leren hoe een programmeertaal wordt omgezet naar machinecode en welke syntax en semantiek ze nodig hebben om te programmeren (Robins, Rountree & Rountree, 2003). Het doel van Grande Omega is om studenten te stimuleren, te activeren en te ondersteunen bij hun theoretische en praktisch leerproces, de leercurve van hun programmeer-vaardigheden te verhogen en de docenten inzicht te bieden in hun leerproces.

De informaticaopleiding probeert meer grip op het leerproces te verwerven door de activiteiten van studenten in Grande Omega te monitoren. Het is echter niet duidelijk of Grande Omega daadwerkelijk bijdraagt aan de gewenste ontwikkeling van programmeerkennis en -vaardigheden (Lathinen, Mutka & Jarvinen, 2005, Milne & Rowe, 2002, Qian & Lehman, 2017). Ook is niet duidelijk welke rol Grande Omega speelt bij het ontwikkelen van computational thinking skills (Brennan & Resnick, 2012, Lye & Koh, 2014, Zhong, Wang, Chen & Li, 2016), een set van probleemoplossende vaardigheden en -houdingen, ingebed in de 21ste eeuwse vaardigheden.

Het doel van het kwantitatief onderzoek naar Grande Omega is om inzicht te verwerven in de relaties tussen inspanning, tijdsbesteding en tentamenresultaten van studenten in Grande Omega. Hierbij is de invloed van regelmatig oefenen en het maken van fouten meegenomen. Het doel van het kwalitatief onderzoek is om inzicht te verwerven in de ontwikkeling van het denken in computationele concepten en computationele methoden (Fisser & Strijker, 2016, Voogt, Brand-Gruwel & Van Strien, 2017) via uitspraken van geïnterviewde studenten.

De onderzoeksgroep voor het kwantitatief, retrospectief onderzoek bestaat uit 132 eerstejaarsstudenten Informatica van Hogeschool Rotterdam. Het onderzoek is vormgegeven als een in situ quasi-experimenteel design. Hierbij zijn de leeractiviteiten en tentamenresultaten van een bestaande groep in een bestaande leersituatie gevolgd. De onderzoeksgroep voor het kwalitatief onderzoek bestaat uit een sample van negen studenten. Dit onderzoek behelst negen, één op één, open ended interviews over drie thema's: Denken in stappen, Abstraheren en Algoritmisch denken.

De hoofdvraag luidt: *Welke invloed heeft de online leeromgeving Grande Omega op de programmeerprestaties en de ontwikkeling van Computational Thinking Skills van eerstejaars studenten informatica bij Hogeschool Rotterdam?*

Uit de resultaten van het kwantitatief onderzoek blijkt dat studenten hogere tentamenresultaten behalen als ze intensief en regelmatig oefenen in Grande Omega en voldoende tijd besteden aan de oefensessies. Het verband tussen het aantal gemaakte oefeningen in Grande Omega, de tijd die aan de oefeningen is besteed en de prestaties is significant. Hoog presterende studenten geven meer juiste antwoorden en maken tegelijk meer fouten in Grande Omega dan laag presterende studenten. Het lijkt erop dat de hoog presterende studenten van zowel de correcte als de incorrecte antwoorden leren (Grassinger, Scheunpflug, Zeinz & Dresel, 2018). Uit de resultaten van het kwalitatieve onderzoek blijkt dat de invloed van Grande Omega zich het meest laat gelden bij het thema Algoritmisch denken, gevolgd door het thema Abstraheren. De meeste beginners melden dat ze nog onvoldoende vaardigheden hebben om werkende code te kunnen schrijven. Bij het thema Denken in stappen is de invloed het kleinst. Het denken in stappen lijkt vooral voorbehouden aan het oplossen van programmeerproblemen. In relatie tot het onderzoek van Voogt et al. (2017) lijkt Grande Omega positieve invloed te hebben op de ontwikkeling van computationele concepten en computationele methoden, maar niet op generieke kennis en -vaardigheden, zoals probleemoplossend vermogen buiten een programmeeromgeving.

Toekomstig onderzoek kan zich richten op het verband tussen het type opdrachten en de leeropbrengsten van studenten, in welke mate deze groep baat heeft bij informeel leren versus formeel leren en hoe het specifiek trainen van CT bij andere cursussen een bijdrage kan leveren aan het hbo-programmeeronderwijs.

Trefwoorden: programmeren voor beginners, feedback, Computational Thinking Skills

What Influence does the Online Learning Program Grande Omega have on Programming Performance and Computational Thinking Skills?

Anne van Brussel

Summary

The computer science program at Rotterdam University of Applied Sciences is looking for innovative ways to teach groups of novice computer scientists to learn how to program.With the online Grande Omega learning program, developed by teachers, all first year students learn how a programming language is converted into machine code and which syntax and semantics they need to program (Robins, Rountree & Rountree, 2003). The aim of Grande Omega is to stimulate, activate and support students in their theoretical and practical learning process, to increase the learning curve of their programming skills and to offer teachers insight into their learning process.

The computer science program tries to gain more control over the learning process by monitoring the activities of students in Grande Omega. However, it is not clear to what extent Grande Omega actually contributes to the desired development of programming knowledge and skills (Lathinen, Mutka & Jarvinen, 2005, Milne & Rowe, 2002, Qian & Lehman, 2017). It is also not clear what role Grande Omega plays in the development of computational thinking skills (Brennan & Resnick, 2012, Lye & Koh, 2014, Zhong, Wang, Chen & Li, 2016), a set of problem solving skills and attitudes, embedded in 21st century skills.

The aim of quantitative research on Grande Omega is to gain insight into the relationships between effort, time allocation and exam results of students in Grande Omega. This includes the influence of regular practice and making mistakes.The aim of qualitative research is to gain insight into the development of thinking in computational concepts and computational methods (Fisser & Strijker, 2016, Voogt, Brand-Gruwel & Van Strien, 2017) through statements from interviewed students.

The research group for quantitative, retrospective research consists of 132 first-year computer science students from Rotterdam University of Applied Sciences. The research is designed as an in situ quasi-experimental design. The learning activities and exam results of an existing group in an existing learning situation have been followed. The research group for qualitative research consists of a sample of nine students. This research involves nine, one on one, open ended interviews on three themes: Thinking in steps, Abstracting and Algorithmic thinking. The main question is: *What influence does the online learning environment Grande Omega have on the programming performance and the development of Computational Thinking Skills of first-year computer science*

students at Rotterdam University of Applied Sciences?

The results of the quantitative study show that students achieve higher exam results if they practice intensively and regularly in Grande Omega and spend sufficient time on the practice sessions. The relationship between the number of exercises performed in Grande Omega, the time spent on the exercises and exam results is significant. High-performing students give more correct answers and, at the same time, make more mistakes in Grande Omega than low-performing students. It seems that the high-performing students learn from both the correct and incorrect answers (Grassinger, Scheunpflug, Zeinz & Dresel, 2018). The results of the qualitative study show that the influence of Grande Omega is most evident in the Algorithmic theme, followed by the Abstraction theme. Most beginners report that they still have insufficient skills to write working code. The influence on the theme Thinking in steps is smallest. Thinking in steps seems to be primarily reserved for solving programming problems. In relation to the research by Voogt et al. (2017), Grande Omega seems to have a positive influence on the development of computational concepts and computational methods, but not on generic knowledge and skills, such as problem-solving ability outside a programming environment.

Future research can focus on the relationship between the type of assignments and the learning outcomes of students, to what extent this group benefits from informal learning versus formal learning and how specifically training CT in other courses can contribute to higher professional education in applied sciences.

Keywords: learning to program, online feedback, Computational Thinking Skills

1. Inleiding

Het nijpend tekort aan software specialisten in een steeds sneller digitaliserende maatschappij onderstreept de economische belangen van adequaat programmeeronderwijs. Bij de Nederlandse hogescholen worden elk collegejaar meer informaticastudenten opgeleid dan het jaar daarvoor.

Hbo-studenten die voor informatica kiezen, verwachten een eigentijds leerprogramma dat aan hun leerbehoeften voldoet. De informaticaopleiding van Hogeschool Rotterdam zoekt daarom naar efficiënte, effectieve en innovatieve manieren om grote groepen beginnende informatici te leren programmeren en hun vorderingen te monitoren.

Enkele docenten van de informaticaopleiding van Hogeschool Rotterdam hebben het online leerprogramma Grande Omega ontwikkeld, waarmee alle studenten vanaf het eerste jaar van hun opleiding leren hoe een programmeertaal wordt omgezet naar machinecode en welke syntax en semantiek ze nodig hebben om te leren programmeren. Grande Omega is een generieke *web-based code interpreter*. Grande Omega wordt via *prototyping* cyclisch in de praktijk getoetst met groepen studenten. Het doel van het programma is om studenten te stimuleren, te activeren en te ondersteunen bij hun theoretische en praktisch leerproces, de leercurve van hun programmeervaardigheden te verhogen en de docenten inzicht te bieden in hun leerproces. De doelgroep van Grande Omega bestaat uit jongvolwassenen die als toekomstige professionals tijdens of na hun studie als programmeur of softwareontwikkelaar in het bedrijfsleven aan de slag gaan.

Grande Omega heeft tijdens het eerste collegejaar een centrale plaats binnen de leerlijn Software Development (DEV). Aangezien beginners vaak fouten maken in de syntax (Milne & Rowe, 2002, Kelleher and Pausch, 2005, in Qian & Lehman, 2017) oefenen studenten zelfstandig in het herkennen, interpreteren en schrijven van kleine stukken code. Ze vullen bestaande programma's aan en breiden deze stapsgewijs uit, waarbij ze hun antwoorden met de verwachtingen van het systeem kunnen vergelijken. Grande Omega voorziet de studenten van uitgestelde feedback door de ingevoerde gegevens als goed of fout te markeren en bij een fout antwoord enkele minuten bedenktijd te geven, voordat de oefening opnieuw wordt aangeboden. Ook verwijst het programma naar interne hulpbronnen. Daarnaast kunnen de studenten bij de werkcolleges van de leerlijn DEV klassikale instructie krijgen van de docenten en onder begeleiding oefenen in het programma. Elke informaticadocent hamert op een benodigd aantal "vlieguren". De opleiding neemt aan dat studenten die regelmatig en intensief oefenen met programmeeropdrachten - ook in hun vrije tijd - hun leerdoelen eerder behalen dan de studenten die dat nalaten.

1.1 Probleemschets

De informaticaopleiding probeert meer grip op het leerproces te verwerven door de inspanning en tijdsbesteding van studenten in Grande Omega te monitoren en hun leerproces bij te sturen tijdens de practicumlessen. Het is echter niet duidelijk of Grande Omega daadwerkelijk bijdraagt aan de gewenste ontwikkeling van programmeerkennis en -vaardigheden. Ook is niet duidelijk welke rol Grande Omega speelt bij het ontwikkelen van Computational Thinking Skills (CT), een set van probleemoplossende vaardigheden en -houdingen, ingebed in de 21ste eeuwse vaardigheden.
In welke mate het oefenen in Grande Omega tot hogere tentamenprestaties leidt en welke invloed het programma op de ontwikkeling van computationele concepten en -methoden heeft, vormt het onderwerp van dit onderzoek.

1.2 Doelstelling

Het doel van het onderzoek naar Grande Omega is om inzicht te verwerven in de relaties tussen inspanning, tijdsbesteding en tentamenresultaten van studenten in Grande Omega. Hierbij wordt de invloed van regelmatig oefenen en het maken van fouten meegenomen. Tevens kunnen de uitspraken van studenten licht werpen op aspecten als beheersing van syntax en semantiek, het toepassen van het geleerde, de aard van de feedback en de ontwikkeling van het denken in computationele concepten en computationele methoden.

Het onderzoek naar Grande Omega sluit aan op het onderzoeksprogramma van Technology Enhanced Learning Innovations (TELI) van het Welten-instituut van de Open Universiteit. In eerder onderzoek is gekeken naar de kwaliteit van open online leeromgevingen, geautomatiseerde feedback en learning analytics om online leergedrag te onderzoeken en te voorspellen. Het onderzoek naar de ontwikkeling van computationele concepten en -methoden sluit aan bij het onderzoek naar de ontwikkeling van Computational Thinking Skills in programmeeronderwijs van onder andere Fisser & Strijker (2016) en Voogt, Brand-Gruwel en Van Strien (2017).

1.3 Theoretisch kader

Leren programmeren is een complexe cognitieve vaardigheid (Robins, Rountree & Rountree, 2003, Lathinen, Mutka & Jarvinen, 2005). Beginners staan voor de uitdaging om zich een uitgebreide set aan simultane vaardigheden eigen te maken (Milne & Rowe, 2002). Beginnende programmeurs onderzoeken vrijwel gelijktijdig hoe een programmeertaal wordt omgezet naar machinecode, waar software voor dient, hoe ze programma's kunnen gebruiken en wat ze ermee kunnen bereiken. Dit leerproces gaat niet zonder slag of stoot. Het verwerven van een mentaal model (Driscoll, 2005) van de computeracties tijdens het uitvoeren van programma's vraagt om aanzienlijke inspanning en tijd. Het mentale model dat programmeurs ontwikkelen bestaat uit een combinatie van domeinspecifieke en

vaktechnische kennis, computionele concepten en -methoden, en inzicht in taken met bijbehorende vaardigheden. De computionele concepten en -methoden vallen onder het begrip Computational Thinking Skills. CT zijn kortweg te definiëren als een set van probleemoplossende vaardigheden en houdingen, waar concepten uit de informatica aan ten grondslag liggen (Lye & Koh, 2014, zoals het denken in stappen, het ordenen van informatie, het besef van volgordelijkheden en het modelleren van gegevens (Wing, 2008, Barr & Stephenson, 2011).

Om een rijker beeld te krijgen van de aspecten die bij het verwerven van programmeer-vaardigheden en het ontwikkelen van CT een rol spelen, gaat de volgende sectie dieper op de wetenschappelijke bevindingen in. Daarna volgen de hypotheses en de onderzoeksvraag.

1.3.1 Wetenschappelijke bevindingen

Een programmeur met 10 jaar expertise wordt in het vakgebied als gevorderde beschouwd. Beginners hebben, door de benodigde leertijd, oppervlakkige en beperkt georganiseerde kennis over programmeren (Qian & Lehman,2017). Beginners hebben tevens moeite om relevante en irrelevante kennis te scheiden. Daarnaast vinden ze het moeilijk om relevante kennis daadwerkelijk toe te passen. Ze hanteren een tijdrovende één-voor-één regel-benadering, zonder het programmeerprobleem op te delen in deelproblemen of gebruik te maken van structuren (Winslow, 1996, in Robins et al., 2003). Beginners leren te werken met schema's en abstracte representaties van processen. (Lathinen, Mutka & Jarvinen, 2005, Qian & Lehman, 2017). Ze leren de syntax en semantiek van programmeertalen. Daarnaast leren ze pragmatische vaardigheden zoals het plannen, ontwerpen, ontwikkelen, testen en debuggen. Beginnende programmeurs leren dus - gelijktijdig - overlappende domeinen (Boulay, 1989, in Robins et al., 2003). Soloway en Spohrer (1989, in Robins et al., 2003) postuleren dat het beperkte begrip van programmeertaalconstructies, zoals variabelen, loops en arrays, bij beginners tot verkeerde aannames kan leiden, waardoor ze bij het ontwerpen en testen van software al gauw vastlopen. Veel bugs bij beginners ontstaan bij het samenvoegen van de onderdelen van een programma. Dit zou eerder door een gebrek aan ontwerpvaardigheden komen, dan door misvattingen over taalconstructies. Uit de metastudie van Robins et al. (2003) blijkt verder dat beginnende programmeurs de syntaxis en semantiek van geïsoleerde regels na veel oefenen wel kennen, maar niet weten hoe ze deze functies moeten combineren in valide programma's. Zelfs als ze programmeerproblemen met pen en papier kunnen oplossen, hebben ze moeite met de vertaling van hun oplossing in een computerprogramma (Winslow, 1996, in Robins et al., 2003).

Intensieve feedback is bij leren programmeren noodzakelijk; verkeerde aannames zijn immers gauw gemaakt. Feedback binnen een online leeromgeving kan direct of uitgesteld worden aangeboden. Uitgestelde feedback zou studenten kunnen stimuleren om te reflecteren op het onjuiste antwoord, waarna ze opnieuw een poging kunnen wagen. Of deze benadering effectief is staat in de literatuur ter

discussie. Van der Kleij, Feskens & Eggen (2015) halen in hun reviewstudie voor- en tegenstanders van uitgestelde feedback aan. Het tijdstip, de aard en de vorm van feedback blijken onlosmakelijk met elkaar verbonden (Anderson, Conrad & Corbett, 1989, Mory, 2004, Hattie, 2009, Kleij et al., 2015). De aard van de feedback heeft invloed op de wijze waarop een student het maken van fouten ervaart (Hattie & Timperley, 2007). Fouten kunnen onzekerheid in de hand werken en prestaties negatief beïnvloeden (Anderson et al., 1989, Mathan, 2005). Studenten met hogere cognitieve vaardigheden zouden meer van fouten leren (Grassinger, Scheunpflug, Zeinz & Dresel, 2018).

Een complex en tijdrovend proces als leren programmeren vraagt om een actieve leerhouding, waarbij zelfeffectiviteit en zelfregulering (Bandura, 1977, 1993, Pintrich 2004, Driscoll, 2005) een belangrijke rol spelen. Zelfeffectieve en zelfregulerende studenten willen de lesstof beheersen, ze leveren meer inspanningen, doen extra oefeningen, informeren zich via online tutorials en corrigeren zichzelf op vermijdingsgedrag (Abara & Lokena, 2010, in Komarraju & Nadler, 2013, Zimmerman, 1990). Deze studenten ontplooien meer leeractiviteiten en zouden daardoor de ontwikkeling van hun Computational Thinking Skills kunnen versnellen.

Volgens Wing (2006) staan beginnende programmeurs voor de uitdaging om in meervoudige abstracties te denken; één van de CT-concepten. Wing (2008) definieert CT als: "de denkprocessen die betrokken zijn bij het formuleren van problemen en hun oplossingen, zodat de oplossingen worden gerepresenteerd in een vorm die effectief kan worden uitgevoerd door een informatieverwerkend systeem". De Computer Science Teachers Association en de International Society for Technology in Education (CSTA & ISTE, 2009) hanteren een zeer uitgebreide definitie: "CT is een probleemoplossend proces dat bestaat uit verschillende kenmerken: a) het formuleren van problemen op een wijze die ons in staat stelt om een computer en andere hulpmiddelen te gebruiken om ze op te lossen, b) het logisch organiseren en analyseren van gegeven, c) het representeren van gegevens door middel van abstracties zoals modellen en simulaties, d) het automatiseren van oplossingen door middel van algoritmisch denken als een reeks geordende stappen, e) het identificeren, analyseren en implementeren van mogelijke oplossingen met als doel om de meest efficiënte en effectieve combinatie van stappen en middelen te bereiken, en f) het generaliseren en overbrengen van dit probleemoplossingsproces naar een breed scala aan problemen". Ondanks de verschillende definities van CT worden abstraheren en analytisch, probleemoplossend denken als de centrale kenmerken van CT beschouwd (Wing, 2008, Barr & Stephenson, 2011). Dit zijn vaardigheden die eveneens in het programmeeronderwijs centraal staan. CT is breder dan programmeren, maar aspecten van programmeren, zoals abstraheren en probleemoplossen, kunnen een bijdrage leveren aan de ontwikkeling van CT (Voogt et al., 2017).

Aangezien er in de loop der jaren behoefte ontstond aan het meten van CT, introduceerden Brennan & Resnick (2012) het Computational Thinking Framework, waarin drie perspectieven op CT

centraal staan. Lye & Koh (2014) en Zhong, Wang, Chen & Li (2016) bouwden hier dankbaar op voort om vanuit een theoretisch perspectief de ontwikkeling van CT bij kinderen in het basisonderwijs en jongvolwassenen in het voortgezet onderwijs te onderzoeken. De belangrijkste drie concepten van het Computational Thinking Framework (Brennan & Resnick, 2012, Lye & Koh, 2014, Zhong et al., 2016) zijn computionele concepten, computionele methoden en computionele perspectieven. Ontwikkelaars gebruiken concepten tijdens het programmeren, zoals sequenties, loops, events, condities of parallellisme. Voorbeelden van methoden zijn incrementeel en iteratief werken, testen en debuggen, remixen, abstraheren en modulariseren. Perspectieven bestaan uit de denkbeelden die ontwikkelaars vormen over zichzelf en de wereld om hen heen, zoals zich uiten en werkgerelateerde connecties maken met vakgenoten met als doel om kennis te delen. Volgens Mayer, Dyck, & Vilberg (1989, in Robins, 2003) is er geen bewijs dat leren programmeren de algemene probleemoplossende vaardigheden van een student verbetert. Wel zou programmeren het leren van nauw verwante vaardigheden stimuleren. Voogt et al.(2017) vonden in hun reviewstudie echter een voorzichtig positief effect van programmeeronderwijs op computionele concepten, computionele methoden en generieke kennis en vaardigheden, zoals probleemoplossend vermogen. Het onderzoek naar de effecten van programmeeronderwijs op CT staat nog in de kinderschoenen, aldus Voogt et al. (2017). In het basis- en middelbaar onderwijs zijn uiteenlopende studies verricht die echter niet alle even betrouwbaar bleken. In het hoger onderwijs is nog weinig onderzoek naar CT verricht (Czerkawski & Lyman, 2015).

1.3.2 Hypotheses

De wijze waarop een informaticaopleiding haar studenten het beste kan ondersteunen bij het leren programmeren blijft onderwerp van wetenschappelijk debat (Lathinen et al., 2005, Milne & Rowe, 2002, Qian & Lehman, 2017). Uit de literatuur (Robins et al., 2003) blijkt dat het verwerven van complexe cognitieve vaardigheden niet alleen inspanning, maar ook leertijd vraagt. De DEV-docenten van de informatica-opleiding van Hogeschool Rotterdam gaan ervan uit dat studenten die intensief en regelmatig oefenen met programmeeropdrachten beter presteren dan studenten die dit niet doen. Studenten die vaak oefenen, zouden logischerwijs ook veel tijd aan de oefeningen besteden, tenzij de oefeningen zo eenvoudig zijn, dat tijd een kleine rol van betekenis speelt. Om die reden is het interessant om te verkennen of er een correlationeel verband is tussen het aantal gemaakte oefeningen, de tijd die aan deze oefeningen is besteed en de hoogte van de tentamenresultaten. De eerste hypothese verkent de verschillen tussen de hoog-en laag presterende studenten: *H1: Hoog presterende studenten maken meer oefeningen en besteden meer tijd aan de oefeningen dan laag presterende studenten.* De tweede hypothese onderzoekt het correlationeel verband: *H1: Er is positieve samenhang tussen het aantal gemaakt oefeningen in Grande Omega, de tijd die aan de oefeningen is besteed en de*

prestaties, uitgedrukt in tentamenresultaten. Waarbij direct in ogenschouw moet worden genomen dat er ook enkele studenten zijn die geheel buiten Grande Omega om leerstrategieën gebruiken om hun kennis te verhogen. Ze bekijken tutorials op YouTube, volgen MOOC's of werken in hun vrije tijd bij een IT-bedrijf. Dit blijkt uit de oriënterende gesprekken die met begeleidende hogerejaarsstudenten (peer coaches) zijn gevoerd. In het vooronderzoek in bijlage 1 zijn deze gegevens uitgewerkt. 95% van de eerstejaarsstudenten volgt echter het aanbod van Grande Omega.

Het uitgangspunt van de ontwikkelaars van Grande Omega is dat studenten leren door veel te oefenen met steeds moeilijkere opdrachten. Ze maken daarbij vanzelfsprekend fouten. In welke mate draagt het maken van fouten in Grande Omega bij aan de leeropbrengsten, uitgedrukt in de hoogte van de tentamenresultaten? Met andere woorden: halen studenten die meer onjuiste antwoorden in het oefenprogramma van Grande Omega geven hogere tentamencijfers dan studenten die meer juiste antwoorden geven? De derde hypothese luidt: *H1: Studenten die veel fouten maken in Grande Omega presteren beter dan studenten die weinig fouten maken.*

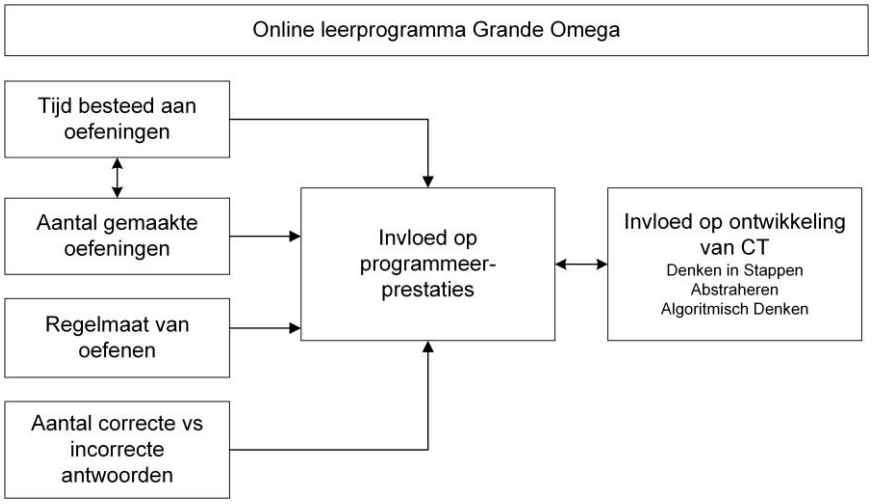
Het lijkt voor de hand te liggen dat zelfeffectieve en zelfregulatieve studenten regelmatig oefenen in Grande Omega en dus in elke onderwijsperiode actief zijn. Zij zouden dientengevolge hogere tentamenresultaten behalen dan studenten die dat niet doen. De vierde hypothese luidt: *H1: Hoog presterende studenten doen in elke onderwijsperiode meer oefensessies in Grande Omega dan laag presterende studenten.*

Studenten zouden door het oefenen in Grande Omega hun Computational Thinking Skills kunnen ontwikkelen (Voogt, Brand-Gruwel & Van Strien, 2017). Beginnende programmeurs kunnen relevante en irrelevante kennis niet altijd scheiden (Winslow, 1996, in Robins, 2003) en ze hebben moeite om programmeerproblemen op te delen in deelproblemen. Aangezien docenten en studenten in het vooronderzoek het belang van de drie vaardigheden Denken in stappen, Abstraheren en Algoritmisch denken benadrukken, zijn uit het Computational Thinking Framework (Brennan & Resnick, 2012, Lye & Koh, 2014, Zhong et al., 2016) de computionele concepten en -methoden richtinggevend geweest voor de CT-vaardigheden die bij de hbo-opleiding Informatica zijn onderzocht. De drie vaardigheden Denken in stappen, Abstraheren en Algoritmisch denken maken overlappend deel uit van computationele concepten en -methoden. Hieruit vloeit de deelvraag voort: *Welke invloed heeft het gebruik van de leeromgeving Grande Omega op de ontwikkeling van Computational Thinking Skills?*

1.3.3. Hoofdvraag en conceptueel model

Het conceptueel model (figuur 1) van het onderzoek geeft de variabelen en de te verwachten relaties en invloeden op het resultaat weer. De hoofdvraag van het onderzoek luidt: *Welke invloed heeft de*

online leeromgeving Grande Omega op de programmeerprestaties en de ontwikkeling van Computational Thinking Skills van eerstejaars studenten informatica bij Hogeschool Rotterdam?



Figuur 1

Conceptueel model van het onderzoek naar de invloed van Grande Omega op programmeerprestaties en de ontwikkeling van Computational Thinking Skills

2. Methode

Na de probleemstelling, doelstelling, hypotheses en vraagstelling is in deze sectie de methode beschreven. Achtereenvolgens komen onderzoeksgroep, ontwerp, materialen, procedures en analyses aan bod.

2.1 Onderzoeksgroep

De populatie bestaat uit 293 mannelijke en vrouwelijke studenten, tussen de 17 en 25 jaar. Deze studenten zijn in september 2017 aan hun voltijdstudie Informatica begonnen bij Hogeschool Rotterdam. Ze vormen gezamenlijk de eerstejaars klassen van de opleiding Informatica van collegejaar 2017-2018. De onderzoeksgroep bestond in eerste instantie uit 140 studenten. Zij gaven schriftelijk toestemming voor kwantitatieve dataverzameling. Van deze groep gaven 44 studenten toestemming voor zowel kwantitatieve als kwalitatieve dataverzameling. Uiteindelijk was van 132 studenten kwantitatieve data beschikbaar voor statistische analyses. Met negen van de 44 studenten is kwalitatief onderzoek uitgevoerd.

2.2 Ontwerp

Het onderzoek is vormgegeven als een in situ quasi-experimenteel design. Hierbij zijn de leeractiviteiten en tentamenresultaten van een bestaande groep in een bestaande leersituatie gevolgd. In deze leersituatie is de invloed gemeten van de onafhankelijke variabelen op de afhankelijke variabelen. Voor het kwantitatief onderzoek bestaan de onafhankelijke variabelen uit a) tijd besteed aan oefeningen, b) het aantal gemaakte oefeningen, c) de regelmaat van oefenen en d) het aantal correcte versus incorrecte antwoorden. De afhankelijke variabele bestaat uit de scores op de DEV-tentamens. Voor het kwalitatief onderzoek is via open ended, één-op-één interviews gemeten welke invloed de leeractiviteiten in Grande Omega hebben op de ontwikkeling van drie CT-concepten en -methoden. De afhankelijke variabele bestaat uit de scores op de niveaus High (H), Medium (M) en Low (L). In figuur 1 zijn de relaties weergegeven in een conceptueel model en in tabel 1 zijn de variabelen en instrumenten weergegeven.

Tabel 1

Overzicht van variabelen en instrumenten. DEV staat voor de leerlijn Development. GO staat voor het leerprogramma Grande Omega. CT staat voor Computational Thinking Skills

Onderzoeksgroep	**Onafhankelijke variabelen**	**Afhankelijke variabelen**	**Meetinstrument**
Ca. 150 studenten Informatica leerjaar 1 Collegejaar 2017-2018 van Hogeschool Rotterdam	a) datum en tijd van een oefensessie in GO b) het aantal oefensessies in GO c) het aantal correcte en incorrecte antwoorden bij een oefensessie in GO	Scores op resultaat van summatieve tentamens DEV	Data log files in GO Tentamencijfers DEV
Sample van max. 10 studenten uit onderzoeksgroep	Concepten en -methoden (CT): a) Denken in stappen b) Abstraheren c) Algoritmisch denken	Score op niveaus H, M en L via kwalitatieve uitspraken	Open ended, één-op-één interviews

Voor de analyse van data is voor retrospective research gekozen. Er is een post hoc-split analyse uitgevoerd van data van leeractiviteiten van de bestaande groep eerstejaarsstudenten in onderwijsperiode 1 t/m 4 in het programma Grande Omega 2017-2018, in combinatie met tentamenresultaten van de leerlijn Development (DEV) over dezelfde periode. Aangezien alle studenten hetzelfde onderwijs volgen en dezelfde ervaringen ondergaan, was een controlegroep niet van toepassing.

De data die in de *backlog* van Grande Omega beschikbaar zijn voor statistische analyse bestaan uit a) datum en tijd van een gemaakte oefensessie, b) het aantal oefensessies dat een student uitvoert en c) het aantal correcte en incorrecte antwoorden bij een oefensessie. Met behulp van de data zijn de hypotheses te onderzoeken om een antwoord te kunnen formuleren op de vraag welke invloed Grande Omega heeft op programmeerprestaties, uitgedrukt in tentamenresultaten.

De drie CT-concepten zijn via open ended interviews geëxploreerd. De drie onafhankelijke variabelen bestaan uit Denken in stappen, Abstraheren en Algoritmisch denken. De afhankelijke variabele bestaat uit de score van de student op de niveaus High (H), Medium (M) en Low (L) per CT-thema. In het vooronderzoek in bijlage 1 zijn de CT-thema's, de bijbehorende rubrics en de scoringsniveaus uitgewerkt en onderbouwd. In bijlage 2 is het interviewprotocol opgenomen.

Voor het onderzoek naar de CT-concepten is een selecte steekproef uit de onderzoekspopulatie genomen. Hierbij is gebruik gemaakt van maximal variation sampling. De steekproef is select door de vrijwillige deelname van de studenten. De doelgerichte sampling bij de kwalitatieve analyse is toegepast om de centrale verschijnselen te begrijpen. Bij het genereren van een theorie over de CT-vaardigheden van de studenten ten opzichte van de leermethode, zijn de uitspraken van de studenten leidend. Aangezien de informatie op *self-report* van de studenten is gebaseerd, en het kwalitatief onderzoek interpreterend van aard is, zijn de resultaten vanuit het perspectief van de onderzoeker weergegeven. De onderzoeker heeft de antwoorden van de geïnterviewde studenten op de themavragen opgenomen, getranscribeerd en vervolgens gecodeerd door typische CT-uitspraken te onderstrepen. Daarnaast is de kern van het betoog in de rechterkolom van het transcript samengevat. Via een rubric met indicatoren zijn de antwoorden in niveaus High (H), Medium (M) en Low (L) ingedeeld. De betekenis van de niveaus van de rubric zijn in het vooronderzoek beschreven en toegelicht. In een vergelijkingstabel (tabel 6) zijn per CT-thema en per niveau exemplarische uitspraken van studenten weergegeven.

2.3 Materialen

Via twee verschillende methoden is data verzameld. Om de hypotheses te kunnen aannemen of verwerpen, is data van studentactiviteiten in Grande Omega opgevraagd bij de docent/eigenaar van het leerprogramma. Deze data behelst de tijd en datum tussen log entry en log out van een sessie, het aantal afgemaakte sessies en het aantal correcte en incorrecte antwoorden bij de oefeningen. Bij de administratie van het opleidingsinstituut, waar het studentvolgsysteem Osiris in beheer is, zijn tentamencijfers van DEV uit onderwijsperiode 1 t/m 4 opgevraagd. Uit beide systemen is alleen data opgevraagd van studenten die schriftelijk toestemming hadden gegeven. De studentnamen en studentcodes zijn geanonimiseerd voordat analyse plaatsvond, conform het positief advies van de commissie ethische toetsing onderzoek.

Om de deelvraag over CT-concepten te kunnen beantwoorden, is data verzameld via negen open interviews. De één op één interviews vonden plaats in een geluidsarme kamer in het gebouw van de opleiding en werden vooraf gegaan door een interviewprotocol (bijlage 2) waarbij het doel van de studie, de tijd om het interview af te ronden en de wijze van uitwerken waren verhelderd. Het interview bestond uit open vragen, waar open reacties op volgden. Project 3 van onderwijsperiode 3 was als leidraad voor de vragen gekozen, omdat in de studenten tijdens dit project bij uitstek hun CT-vaardigheden konden toepassen. De vragen waren gerubriceerd in drie CT-thema's: 1) Denken in stappen, 2) Abstraheren en 3) Algoritmisch denken. Denken in stappen is gedefinieerd als a) een probleem opdelen in kleinere deelproblemen of in deelvragen, en b) een probleem zo formuleren dat het met behulp van een computer is op te lossen. Abstraheren is gedefinieerd als a) de essentie verduidelijken zonder zich in details te verliezen, en b) schematiseren/modelleren door gebruik te maken van schetsen, tabellen, grafieken of modellen. Algoritmisch denken is gedefinieerd als a) stap-voor-stap specifieke en expliciete instructies maken om een proces uit te voeren, en b) logische volgordelijkheid toepassen. Voorafgaand aan de themavragen zijn aan de studenten algemene vragen gesteld over vooropleiding en programmeerervaring. Daarnaast is de visie van de geïnterviewden op Grande Omega vastgelegd. De drie thema's zijn na literatuurstudie tijdens het vooronderzoek in overleg met docenten en peercoaches vastgesteld. Het vooronderzoek is toegelicht in bijlage 1. De namen van de thema's waren in het interviewprotocol door de codes DIS, AB en AD vervangen, zodat de geïnterviewde geen hints kreeg in welke richting zijn/ haar antwoorden zouden moeten leiden.

2.4 Procedure

In september 2017 heeft het eerste overleg plaatsgevonden over een mogelijk onderzoek naar Grande Omega. Dit overleg vond plaats bij het IT-bedrijf, waar de docent/eigenaar als *chief technical officer* werkzaam is. Het gesprek met de docent en de directeur - die tevens eigenaar is van Grande Omega - vormde het vertrekpunt van het onderzoek. Na overleg met de directeur, docent en de thesisbegeleider volgde in oktober 2017 een verkennend gesprek met de opleidingsmanager van Informatica en de kwaliteitscoördinator van het instituut. De opleidingsmanager gaf toestemming voor het onderzoek en de kwaliteitsmanager adviseerde bij de onderzoeksopzet.

Het vooronderzoek met informaticadocenten, peer coaches en studiebegeleiders vond plaats in het voorjaar van 2018. De gesprekken met drie informaticadocenten en vier peer coaches ondersteunden het iteratieve proces bij het vormgeven van het kwalitatief onderzoek. De studiebegeleiders en peer coaches gaven informatie over de inhoud van de projecten. De concept-themavragen zijn daarna door twee afstudeerstudenten uit het vierde jaar doorgenomen. Ze hebben aanbevelingen gedaan om de vragen begrijpelijker en concreter te maken voor de studenten uit het eerste jaar. Alle vragen zijn vervolgens toegespitst op de praktijkervaringen van studenten in het eerste

leerjaar en de semantiek die zij hanteren. De peer coaches hielpen bij het bedenken van mogelijke antwoorden op de vragen, om het herkennen van signaalwoorden te bevorderen. Eén van de peercoaches controleerde de gebruikte terminologie. Na drie reviews van de concept-interviewvragen en de CT-thema's met bijbehorende indicatoren en niveaus, kregen de instrumenten definitief vorm.

Het onderzoek heeft tussen 22 april en 28 juli 2017 in onderwijsperiode 4 plaatsgevonden. Alle 293 studenten hebben twee weken voor het verzamelen van de toestemmingsformulieren per mail een Nederlandstalige en Engelstalige brief ontvangen met daarin de aankondiging van het onderzoek. In deze brief is de eerder verkregen institutionele toestemming vermeld en is nadruk gelegd op de bescherming van anonimiteit van deelnemers. Op advies van de docenten, studiebegeleiders en peer coaches zijn de studenten twee weken na de brief persoonlijk in hun klassen benaderd voor schriftelijke toestemming. Toestemming vragen via de digitale weg zou volgens de adviseurs tot een lagere response leiden. Er is gekozen voor een toestemmingsformulier waarop de student zowel voor het kwantitatieve als het kwalitatieve onderzoek toestemming kon geven. Op deze manier kon de onderzoeker efficiënter studenten benaderen die aan de interviews wilden meewerken. Van de 293 studenten zijn 153 studenten persoonlijk bereikt in hun eigen klassen. De studiebegeleiders hebben op verzoek van de onderzoeker de lijst met 293 studenten doorgenomen en verklaard dat een deel van de studenten inactief of uitgeschreven was. Studenten nemen om uiteenlopende redenen gedurende het eerste jaar afscheid; het gevraagde werk- en denkniveau is te hoog of te laag, hun interesses zijn gewijzigd, de studie sluit niet aan bij hun verwachtingen of hun gezondheid en persoonlijke omstandigheden zijn veranderd. Om toevallige afwezigheid in de klas uit te sluiten, heeft de onderzoeker alle klassen een week later voor de tweede keer bezocht en bij de docenten toestemmingsformulieren achtergelaten. De docenten gaven studenten de gelegenheid om tijdens de les het informed consent formulier in te vullen. 140 studenten gaven schriftelijk toestemming voor kwantitatieve dataverzameling in Grande Omega en het studentvolgsysteem Osiris, waarin de tentamenresultaten zijn vastgelegd. 13 studenten gaven geen toestemming. Tien studenten leverden hun toestemmingsformulier te laat in, na de uitvoering van het onderzoek.

Aan het begin van het nieuwe collegejaar 2018-2019 waren de data in Grande Omega en Osiris compleet. De data van Grande Omega zijn in september 2018 opgevraagd en in oktober 2018 ontvangen, aangezien de eigenaren van Grande Omega nog software moesten schrijven om de gevraagde gegevens zodanig op te halen uit het systeem, dat analyse mogelijk was. Daarna volgde een review van de verkregen data, waarbij *outliers* en ontbrekende gegevens in overleg met de docent/eigenaar en een collega/data-analist zijn geïnspecteerd. De gegevens van acht studenten die geen activiteiten in Grande Omega verrichten of tussentijds waren uitgeschreven, zijn verwijderd. De data van de onderzoekseenheden (N = 132) zijn met behulp van de collega/data-analist via de programmeertaal Python naar een Excelbestand geconverteerd, zodat de onderzoeker deze data na

controle in SPSS kon analyseren. De data uit Osiris is rechtstreeks in een Excelbestand aangeleverd en toegevoegd aan het SPSS-bestand.

44 van de 140 studenten gaven toestemming voor zowel kwantitatieve als kwalitatieve dataverzameling. De onderzoeker heeft van deze 44 studenten de tentamencijfers van de vakken Analyse en Development in de eerste tot en met de derde onderwijsperiode opgevraagd en het gemiddelde cijfer berekend. De cijfers van Analyse zijn alleen gebruikt bij de selecte steekproef uit de onderzoeksgroep om maximal variation sampling toe te kunnen passen, aangezien op dat moment onvoldoende tentamencijfers voor DEV beschikbaar waren om inzicht te krijgen in de gemiddelde resultaten. Cijfers van herkansingen zijn hierbij buiten beschouwing gelaten, want herkansingen vinden vaak later in het collegejaar plaats. De 44 studenten zijn daarna gelabeld met code AA, BA of A. Code AA staat voor *Above Average*: het gemiddeld cijfer voor de tentamens van de vakken Analyse en Development is dan gelijk aan of hoger dan 7,5. Code BA staat voor *Below Averag*e: het gemiddeld cijfer is gelijk aan of lager dan 5,5. Code A staat voor *Average*: Het gemiddeld cijfer ligt tussen een 5,5 en 7,5.

De steekproef voor de kwalitatieve dataverzameling was op 10 studenten gebaseerd, aangezien verwacht werd dat de uitspraken van deze sample voldoende representatief zou zijn voor de volledige onderzoeksgroep. Van de 44 studenten zijn 28 verschillend gelabelde studenten via maximal variation sampling uitgenodigd voor het interview. De sampling zorgde voor een natuurlijke variatie in leeftijd, achtergrond, geslacht en prestaties. Eén student heeft zich per mail afgemeld en 17 studenten gaven geen response, ook niet na herhaalde verzoeken per mail. Tien studenten reageerden positief op de uitnodiging. Eén student verscheen niet op het afgesproken tijdstip. Negen studenten hebben uiteindelijk deelgenomen aan het één-op-één interview door de onderzoeker. De negen geïnterviewde studenten bestaan uit twee vrouwen en zeven mannen. Drie studenten hadden ten tijde van het interview label AA, vier het label BA en twee het label A. De labeling ten tijde van het interview is terug te vinden in tabel 7. Voordat de interviews in onderwijsperiode 4 plaatsvonden, gaven de negen studenten schriftelijk toestemming voor het opnemen van het interview en het anoniem verwerken van hun antwoorden in een transcript. Het eerste interview vond plaats op 4 juni 2018, het laatste op 2 juli 2018. Het interview duurde 30 tot 45 minuten. Na het interview is elke student mondeling bedankt voor zijn of haar bijdrage aan het onderzoek.

2.5 Analyses

Op basis van de tentamenresultaten van de onderzoekseenheden is de bestaande groep (N=132) in tweeën gesplitst. Groep AA (N=104), Above Average, presteert hoger bij het tentamen DEV (gemiddeld tentamencijfer = ≥ 5,5) dan groep BA (N=28), Below Average (gemiddeld tentamencijfer = < 5,5). Bij de twee groepen is met een onafhankelijke T-toets nagegaan of de score van de

onafhankelijke variabele op de afhankelijke variabele verschilt. Het standaard statistisch significantieniveau dat hierbij is gehanteerd, bedraagt 0.05. Daarnaast is correlationeel onderzoek gedaan via Pearson's relatiecoëfficiënt *r* naar de onderlinge samenhang tussen de variabelen. Ook hierbij is het standaard significantieniveau van 0.05 gehanteerd.

Bij de analyse van het kwalitatief onderzoek zijn eerst de ervaringen van de geïnterviewde studenten (N=9) met Grande Omega behandeld. Dit om een algemeen beeld te schetsen. Daarna zijn de CT-vaardigheden behandeld, aan de hand van de thema's Denken in stappen, Abstraheren en Algoritmisch denken. De antwoorden van de geïnterviewde studenten zijn per thema in negen transcripts uitgewerkt en via een rubric met indicatoren in niveaus High (H), Medium (M) en Low (L) ingedeeld. De betekenis van de niveaus van de rubric zijn in het vooronderzoek in toegelicht. Exemplarische uitspraken van studenten zijn per CT-thema (Denken in stappen, Abstraheren en Algoritmisch denken) en per niveau (H, M en L) verzameld en in een vergelijkingstabel weergegeven. Vervolgens zijn de scores op de niveaus H, M en L van waarden voorzien: H=3, M=2 en L=1. Door per CT-thema het gemiddelde van de som van de scores te berekenen, is bepaald in welk CT-thema de invloed van Grande Omega zich het duidelijkst manifesteert.

3. Resultaten

3.1 Kwantitatief onderzoek: beschrijving onderzoeksgroep

Op basis van de tentamenresultaten van de onderzoekseenheden is de bestaande groep (N=132) in tweeën gesplitst. Groep AA (N=104), Above Average, presteert hoger bij het tentamen DEV (gemiddeld tentamencijfer = ≥ 5,5) dan groep BA (N=28), Below Average (gemiddeld tentamencijfer = < 5,5). Bij de twee groepen is eerst met een T-toets nagegaan of de score van de onafhankelijke variabele op de afhankelijke variabele verschilt. Daarna is correlationeel onderzoek gedaan via Pearson's relatiecoëfficiënt *r* naar de onderlinge samenhang tussen de variabelen.

3.2 Resultaat statistische analyses

Ten eerste is met een onafhankelijke T-toets onderzocht of AA studenten gemiddeld meer oefeningen in Grande Omega maken dan BA studenten. Uit de toets blijkt dat AA studenten (M = 1969,8, SD = 1273,3) significant meer oefeningen maken dan BA studenten (M = 1075,6, SD = 617,9): t(130) = 3,59, p = < 0,05.

Ten tweede is met een T-toets onderzocht of AA studenten gemiddeld meer tijd besteden aan de oefeningen in Grande Omega dan BA studenten. Uit de toets blijkt dat AA studenten (M = 5163,1, SD = 3914,1) significant meer tijd besteden dan BA studenten (M = 2289,4, SD = 1822,6): t(130) =

5,57, p = < 0,05. De hypothese: *Hoog presterende studenten (tentamencijfer ≥ 5,5) maken meer oefeningen en besteden meer tijd aan de oefeningen dan laag presterende studenten (tentamencijfer < 5,5),* wordt aangenomen.

Ten derde is onderzocht of er onderlinge samenhang is tussen het aantal gemaakte oefeningen in Grande Omega, de tijd die aan de oefeningen is besteed en de prestaties. Het correlationeel onderzoek is met behulp van Pearson's relatiecoëfficiënt *r* uitgevoerd. Uit de analyse blijkt dat er samenhang is: er is een sterk positief en significant verband (.90 -.70) tussen het aantal gemaakte oefeningen en de tijd die aan de oefeningen is besteed: een student die veel oefeningen maakt, besteedt ook veel tijd aan de oefeningen: r =.859, p = 0.00. Daarnaast is er een zwak positief en significant verband (.40 -.10) tussen het aantal gemaakte oefeningen en de hoogte van de tentamencijfers. Gemiddeld behalen studenten die vaak oefenen hogere cijfers dan studenten die minder vaak oefenen. Het verband is significant: r =.184, p = 0.017. Tenslotte is er een zwak positief en significant verband (.40 -.10) tussen de tijd die aan de oefeningen is besteed en de hoogte van het tentamencijfer. Studenten die gemiddeld veel tijd aan de oefeningen besteden, halen hogere cijfers dan studenten die weinig tijd aan de oefeningen besteden: r =.205, p = 0.00. De hypothese: *Er is positieve samenhang tussen het aantal gemaakt oefeningen in Grande Omega, de tijd die aan de oefeningen is besteed en de prestaties, uitgedrukt in tentamenresultaten,* wordt aangenomen. In bijlage 4 is het correlationeel onderzoek statistisch toegelicht.

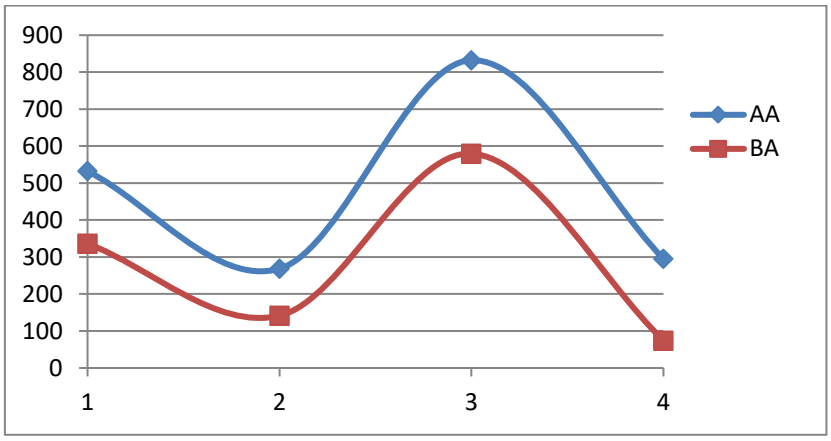
Ten vierde is via een T-toets onderzocht of er verschil is tussen beide groepen in het aantal juiste antwoorden bij de oefeningen en het aantal gemaakte fouten bij de oefeningen.
AA studenten (M = 810,5, SD = 595,8) blijken significant meer correcte antwoorden te geven dan BA studenten (M = 349,9, SD = 221,031): (t(130) = 6,41, p = < 0,05). Dit ligt voor de hand, aangezien AA studenten in totaal meer oefeningen maken dan BA studenten. De AA studenten (M = 1159,3, SD = 733,9) geven echter ook significant meer incorrecte antwoorden bij de oefeningen dan BA studenten(M = 725,7, SD = 425,6): (t(130) = 2,98, p = < 0,05). Het lijkt erop dat de AA studenten leren van zowel correcte als incorrecte antwoorden. De hypothese: *Studenten die veel fouten maken in Grande Omega presteren beter dan studenten die weinig fouten maken,* wordt aangenomen.

Tabel 2

Vergelijkingstabel van gemiddelden van het aantal gemaakte oefeningen per vier onderwijsperiodes. De afkorting Tot exc. staat voor het totaal aantal gemaakte oefeningen.

Group	***Tot exc. Op1***	***Tot exc. Op2***	***Tot exc. Op3***	***Tot.exc. Op4***
AA	533 (N=105, SD=333)	269 (N=108, SD=268)	832 (N=104, SD=832)	295 (N=113, SD=295)
BA	336 (N=27, SD=175)	142 (N=24, SD=141)	580 (N=28, SD=579)	74 (N=19, SD=74)

Ten vijfde is met een vergelijkingstabel van gemiddelden en een T-toets per onderwijsperiode nagegaan of studenten die regelmatig oefenen in Grande Omega - dus in elke onderwijsperiode meerdere oefensessies uitvoeren - hogere tentamenresultaten behalen dan studenten die dat niet doen. In tabel 2 zijn de gemiddelden per groep en per onderwijsperiode weergegeven. In figuur 2 zijn deze gegevens gevisualiseerd.



Figuur 2

Grafische representatie van gemiddelden van het aantal gemaakte oefeningen per onderwijsperiode. Op de Y-as staat het totaal aantal oefensessies, op de X-as staan de vier onderwijsperioden

Uit de tabel blijkt dat AA studenten in alle vier onderwijsperioden meer oefeningen maken dan BA studenten. Dit verschil is voor periode 1 ($t(80) = 4{,}20$, $p = < 0{,}05$.), voor periode 3 ($t(130) = 2{,}30$, $p = < 0{,}05$) en voor periode 4 ($t(121) = 6{,}51$, $p = < 0{,}05$) significant. Voor periode 2 is het verschil niet significant: $t(130) = 1{,}76$, $p = > 0{,}05$. De hypothese: *AA studenten doen in elke onderwijsperiode meer oefensessies in Grande Omega dan BA studenten,* wordt deels aangenomen. AA studenten oefenen in drie onderwijsperiodes significant meer dan BA studenten. In periode 2 maken de AA studenten ook meer oefeningen, maar daar is het verschil niet significant.
In bijlage 5 zijn de T-toetsen toegelicht.

3.3 Samenvatting kwantitatief onderzoek

Samengevat blijkt uit de analyses dat studenten hogere tentamenresultaten behalen als ze intensief en regelmatig oefenen in Grande Omega en voldoende tijd besteden aan de oefensessies. Het verband tussen het aantal gemaakte oefeningen in Grande Omega, de tijd die aan de oefeningen is besteed en de prestaties is significant. Hoog presterende studenten geven meer juiste antwoorden en maken

tegelijk meer fouten in Grande Omega dan laag presterende studenten. Het lijkt erop dat de hoger presterende studenten van zowel de correcte als de incorrecte antwoorden leren.

3.4 Kwalitatief onderzoek: beschrijving onderzoeksgroep

Uit de onderzoeksgroep (N=132) is een selecte groep studenten (N=9) één op één geïnterviewd. Alle studenten hebben eerst hun ervaringen met Grande Omega met de onderzoeker gedeeld en daarna antwoord gegeven op de themavragen. Enkele studenten hebben aanbevelingen gedaan of wensen doorgegeven ten aanzien van de ontwikkeling van Grande Omega als leertool. In Tabel 7 staan de kernuitspraken van de geïnterviewde studenten over Grande Omega. De volledige transcripts van de audio-opnamen zijn als bijlage 3 toegevoegd aan dit onderzoeksrapport.

3.5 Resultaat ervaringen met het leerprogramma

Uit tabel 7 blijkt dat de negen studenten sterk uiteenlopende meningen hebben over de bijdrage van Grande Omega aan hun persoonlijke programmeervaardigheden. ST23 vindt bijvoorbeeld dat Grande Omega voor 100% heeft bijgedragen en geeft op een schaal van 1 t/m 10 ongeveer 4,5 punten stijging van de eigen voortgang ten opzichte van het ingangsniveau op. ST52 zet de bijdrage van Grande Omega op 0% en vindt dat klasgenoten, peer coaches en Internet aan de stijging van een half punt van de eigen voortgang hebben bijgedragen. ST52 heeft ruime IT-ervaring en vindt dat leren in Grande Omega vooral bestaat uit het oplossen van code van Grande Omega:

"Je bent meer bezig met hoe Grande Omega werkt en wat zij van je verwachten, in plaats van dat je bezig bent met programmeren. Een groot deel van het programmeren is ook dat je creatief kan zijn in hoe je iets maakt en Grande Omega staat dat momenteel niet toe. En dat is heel nadelig."

ST32 ziet Grande Omega als "poortwachter" om te controleren of de code is begrepen.

De overige zeven studenten hebben gemiddeld 45% van hun persoonlijke programmeervaardigheden te danken aan Grande Omega. De overige 55% is te danken aan zelfstudie, practicumlessen DEV, projectonderwijs, klasgenoten en peer coaches. ST69 heeft een mbo-opleiding IT-beheer afgerond en zet de bijdrage van Grande Omega op 50% en 3 punten stijging van de eigen voortgang. Deze student heeft van Grande Omega geleerd wat classes doen en wat code op de achtergrond uitvoert:

"En als iets fout gaat, dan kan je echt stap voor stap door de code heenlopen uit je hoofd en dan weet je oh, daar gaat ie fout. Zo helpt het mee. Ik had vroeger nog wel eens dat ik hele dagen met een probleem zat, en nu veel minder gelukkig".

Geen enkele student ziet een heldere relatie tussen Grande Omega en het projectonderwijs, ook niet wanneer ze het oplossen van problemen voor ogen houden. ST111 zegt hierover:

"Eerlijk gezegd, Grande Omega en project (3) komen letterlijk niet met elkaar overeen. Want bij project (3) moet je toch gebruikmaken van allerlei andere soorten termen en technieken om überhaupt te visualiseren wat je wilt visualiseren."

ST111had geen programmeerervaring en heeft van Grande Omega de basisprincipes van coderen en het maken van classes geleerd. ST68 heeft ook geen IT-ervaring en verwoordt het zo:

"*Nou, als ik een probleem had bij projecten, dan dacht ik niet, goh, laat ik Grande Omega openen om te kijken van eh…of daar de oplossing in zit. Dat kwam niet in me op".*

Vijf van de negen studenten spreken hun behoefte uit aan inhoudelijke feedback in Grande Omega. ST32 zegt hierover:

"Ik heb het liever, dat ik zie dat er meer feedback wordt gezet, want ik merk dat ik constant vastloop. Als ik iets fout heb, heb ik geen idee waarom ik het fout heb en dan moet ik naar een peercoach of een leraar, waardoor het dus niet echt zelfstandig is."

ST52 verwoordt het als volgt:

"Momenteel als je iets invult, dan is het gewoon - bam - fout. Er staat dan fout of goed en je kan niet leren om opdrachten te maken als er alleen maar staat fout of goed…Dan weet je wel dat het fout is, maar je weet niet wat er fout is."

ST69 zegt hierover:

"Dan krijg je heel vaak een vage error code… ik heb geen flauw idee waar het fout gaat...je krijgt feedback maar je hebt geen idee wat het is."

ST108 is positief over Grande Omega als concept, maar vindt het te theoretisch. ST108 had verwacht dat het programma getest zou zijn, aangezien er technische problemen optraden tijdens de tentamens:

"...Stel dat ze echt iets hadden opgeleverd dat heel goed werkte, dan denk ik dat het positief ontvangen werd. Puur omdat het niet goed werkte is het negatief ontvangen en vindt iedereen het eigenlijk slecht. Maar de achterliggende gedachte van het systeem is wel goed."

ST38 adviseert om de syntax van GO te vervangen door de syntax van C#:

"Ik denk dat het beter is als Grande Omega dezelfde syntax gebruikt, van de echte talen. Ik weet wel dat het een doel is om de logica te leren, maar het verschil tussen de syntax van Grande Omega en de echte bestaande talen maakt het soms verwarrend voor de studenten."

3.6 Samenvatting van de ervaringen

De meeste studenten staan welwillend tegenover de ontwikkeling van het leerprogramma, al zijn ze kritisch over de gebruikte syntax, de betrouwbaarheid van het tentamenprogramma, de wijze van feedback en de aansluiting van Grande Omega op het projectonderwijs, vooral wanneer ze hun pas verworven programmeervaardigheden willen toepassen. In de discussie worden deze bevindingen verder uitgediept.

3.7 Resultaat invloed van het leerprogramma op CT

3.7.1 Thema Denken in Stappen

Bij het thema Denken in stappen zijn studenten die voornamelijk op niveau L functioneren, nog niet in staat om de probleemstelling en de aanpak van project 3 zelf te organiseren. Ze denken nog niet in de mogelijkheden van het opdelen van complexe problemen of in een stapsgewijze aanpak. Ze nemen vaak een afwachtende houding aan en leunen op de expertise van teamgenoten.

ST108 verwoordt dit zo:

Het was meer eigenlijk, tijdens het maken…van het programmeren, kwamen we steeds met nieuwe ideeën doordat het net weer een andere richting opging. Eigenlijk was het meer: go-with-the-flow-achtig. We hadden wel een globaal beeld in gedachten maar we konden op elk moment afwijken en met iets beters dat doel bereiken, zeg maar…"

Tabel 3

Scores van de antwoorden van studenten bij het thema Denken in stappen.

Het gemiddelde van de totaalscore:17.3/9 = ***1.92***

(ST23)	(ST69)	(ST52)	(ST32)	(ST38)	(ST65)	(ST68)	(ST111)	(ST108)
LLLL	HHMH	LMML	HHHH	HMHH	HHHH	LLL	LLLL	LML
1-1-1-1	3-3-2-3	1-2-2-1	3-3-3-3	3-2-3-3	3-3-3-3	1-1-1	1-1-1-1	1-2-1
M 1.00	**2.75**	**1.50**	**3.00**	**2.75**	**3.00**	**1.00**	**1.00**	**1.30**

Studenten die voornamelijk op niveau M functioneren, proberen met vallen en opstaan een georganiseerde aanpak te vinden. Ze werken eerst enigszins richtingloos en daardoor inefficiënt. Na het optreden van frictie in het team, toegenomen tijdsdruk of feedback van de docent, herzien ze hun benadering en gaan zaken op een rij zetten, waarna ze meer stapsgewijs aan de slag gaan. Studenten die voornamelijk op niveau H functioneren, weten dat ze het probleem in deelproblemen moeten omzetten om op een efficiënte en effectieve manier resultaat te bereiken. Ze passen een projectstrategie toe en verdelen de taken via een stapsgewijze aanpak, waarbij ze steeds in het oog houden dat ze het probleem zo moeten formuleren dat het met een computer is op te lossen. In tabel 3, 4 en 5 zijn de scores op de niveaus weergegeven die bij de antwoorden van de studenten horen.

Bij het thema Denken in stappen luidt de eindvraag: Op welke wijze helpt Grande Omega bij het oplossen van een probleem (bij project 3)? Studenten, die vinden dat ze van Grande Omega vooral de belangrijkste programmeervaardigheden hebben geleerd, noemen code begrijpen, classes en

functies, strings en types. ST32 en ST69 hebben al programmeerervaring en passen decompositie toe. ST32 verwoordt decompositie als volgt:

"Ik deel het eerst op in subproblemen, want vaak is een probleem heel groot."

ST69 benoemt de voordelen van het denken in stappen:

"*…en dan ga je zo stap voor stap proberen alles op te lossen. Kleine stapjes."*

Deze student herkent ook een stapsgewijze aanpak bij Grande Omega:

"*…als er iets fout gaat (in Grande Omega) dan kan je echt stap voor stap door de code heenlopen."*

ST108 heeft ervaring in programmeren en zegt:

"*..Bijvoorbeeld, ik hak het probleem op in stukjes en probeer ik een klein deel op te lossen.*"

ST52 heeft praktische IT-ervaring, leert nu de theorie, en probeert een probleem zo te formuleren dat het met een computer is op te lossen. Deze student loopt tegen het leeraspect aan dat in de literatuur wordt geschetst: beginners die met syntaxis en semantiek oefenen, weten vaak nog niet hoe ze deze kennis kunnen toepassen (Robins et al., 2003): ST52 zegt hierover:

"Ja, je hebt een deel van de theorie van ja, ik weet hoe ik statements moet maken en dingen moet vergelijken. Maar je weet nog steeds niet hoe je het opschrijft".

ST11 heeft geen IT-ervaring en kan een probleem nog niet zelfstandig in code omzetten:

"…dat is nog steeds een lastig iets…hoe ik dat moet doen, want heel vaak weet ik dat gewoon niet".

ST23 heeft ook geen IT-ervaring en functioneert voornamelijk op niveau L ten aanzien van het denken in stappen bij het project. Deze student vindt echter dat het oefenen in Grande Omega wel degelijk helpt bij het formuleren van een programmeerprobleem:

"…I remembered we had such a thing before in coding in Grande Omega and I tried the basics, what I knew about the basics, about for loop. I used it. I tried to write code in that way. Using for loop and basics and the model that I had in my mind, the pattern I had in my mind from somewhere in Grande Omega. And it worked."

Uit deze uitspraak is voorzichtig af te leiden dat ST23 bezig is om een mentaal model te ontwikkelen.

Samengevat herkent ongeveer de helft van de negen studenten de voordelen van een stapsgewijze aanpak bij het programmeren, maar dat betekent niet dat ze bij de aanpak van een project ook decompositie van een probleem toepassen. Het denken in stappen lijkt vooral voorbehouden aan het oplossen van programmeerproblemen. De meeste beginners die de theorie in Grande Omega hebben geleerd, merken dat zenog onvoldoende vaardigheden hebben om werkende code te kunnen schrijven. Uit tabel 3 blijkt dat het gemiddelde van de totaalscore bij het thema Denken in stappen 1.92 bedraagt op een schaal van 1-3.

3.7.2 Thema Abstraheren

Bij het thema Abstraheren verliezen studenten die op niveau L functioneren, zich in details. Ze kunnen de essentie van het probleem nog niet benoemen. Ze testen niet of nauwelijks. Ze weten nog niet hoe ze met behulp van lijstjes, rubrieken of tabellen informatie als een schema of model kunnen weergegeven. Studenten die op niveau M functioneren, zoeken naar de essentie, maar hebben moeite met het isoleren van relevante informatie. Ze zijn zich bijvoorbeeld bewust van de noodzaak van testen, maar doen dat nog ongestructureerd en ze gebruiken nog geen schematische hulpmiddelen. Studenten die op niveau H functioneren, isoleren relevante informatie en rubriceren noodzakelijke gegevens, zoals requirements. Ze gebruiken schema's of modellen om de informatie te structureren en testen hun deelproducten met behulp van een methode.

ST65 verwoordt dat zo:

"We hadden in ons Trello-board kleurtjes gecombineerd met Must, Could en Should. Must was dan rood, het meest belangrijk, zeg maar de hoofdfunctie…"

Tabel 4

Score van de antwoorden van studenten bij het thema Abstraheren.

*Het gemiddelde van de totaalscore: 18.8/9=**2.08***

(ST23)	(ST69)	(ST52)	(ST32)	(ST38)	(ST65)	(ST68)	(ST111)	(ST108)
MM	MM	LL	MM	HH	HH	MM	ML	MMH
2-2	2-2	1-1	2-2	3-3	3-3	2-2	2-1	2-2-3
M 2.00	**2.00**	**1.00**	**2.00**	**3.00**	**3.00**	**2.00**	**1.50**	**2.30**

De eindvraag van het thema Abstraheren (AB) luidt: Op welke wijze helpt Grande Omega bij het onderscheid maken tussen wat belangrijk is en minder belangrijk (in project 3)? De meeste studenten reageren met een stellig "niet". Ze zien geen relatie tussen het leerprogramma en de vaardigheid om onderscheid te maken in belangrijke en minder belangrijke zaken bij het project.

ST38 zegt hierover:

"Volgens mij heeft Grande Omega niets te maken met onderscheiden binnen de projecten, want Grande Omega is meer gebaseerd op de logica van het programma".

ST68 beperkt zich tot één aspect om de essentie van het project te beschrijven:

"*De belangrijkste onderdelen…de code natuurlijk. Daar draait het allemaal om."*

Enkele studenten maken gebruik van schematische hulpmiddelen zoals tabellen en modellen.

ST38 zet functionaliteiten op een rij:

"*…daarin zetten we de prioriteit voor elke functionaliteit. Hoe belangrijk is de functionaliteit, hoe belangrijk is de code…"*

ST108 beschrijft dat de projectgroep requirements prioriteert:

"We hadden het gebaseerd op het MOSCOW-principe, dus: must have, should have en could have."

ST108 herinnert zich dat abstraheren bij Grande Omega wel aan bod is geweest:

" Problemen aanpakken…hoe je volgens mij dingen abstract maakt…daar bedoelen ze mee, zo eenvoudig mogelijk."

Samengevat vinden zeven van de negen studenten dat Grande Omega nauwelijks tot geen bijdrage levert aan het onderscheiden of schematiseren van informatie. De studenten die gebruik maken van schematische hulpmiddelen, hadden al programmeerervaring voor zij aan de studie begonnen. Uit tabel 4 blijkt dat het gemiddelde van de totaalscore bij het thema Abstraheren 2.08 bedraagt op een schaal van 1-3.

3.7.3 Thema Algoritmisch denken

Bij het thema Algoritmisch denken kunnen studenten die op niveau L functioneren, nog geen logische volgordelijkheid herkennen. Zij zijn nog bezig om zich de syntax en semantiek eigen te maken. Als gevolg daarvan zijn ze nog niet in staat om instructies voor de uitvoering van een proces vorm te geven. Studenten die op niveau M functioneren, maken een aanzet om de stappen voor het uitvoeren van een proces in instructies om te zetten. Zij ontwikkelen inzicht in de logica. Studenten die op niveau H functioneren, maken stap-voor-stap specifieke en expliciete instructies om een proces uit te voeren en doen dat in een logische volgorde.

Tabel 5

Scores van de antwoorden van studenten bij het thema Algoritmisch denken.

*Het gemiddelde van de totaalscore: 20.4/9=**2.26***

(ST23)	(ST69)	(ST52)	(ST32)	(ST38)	(ST65)	(ST68)	(ST111)	(ST108)
HML	HHMH	MMLH	HHM	HHH	HHMM	LLML	LMLM	HMHH
3-2-1	3-3-2-3	2-2-1-3	3-3-2	3-3-3	3-3-2-2	1-1-2-1	1-2-1-2	3-2-3-3
M 2.00	**2.75**	**2.00**	**2.66**	**3.00**	**2.50**	**1.25**	**1.50**	**2.75**

De eindvraag van het thema luidt: Op welke wijze helpt het oefenen in Grande Omega (bij een probleem omzetten in code en zorgen dat code begrijpelijk is)?

ST32 zegt hierover:

"*De manier waarop ik Grande Omega ervaar is dat het je leert om code van een ander te begrijpen…het leert me niet hoe ik zelf code moet schrijven."*

ST65 zegt:

"…je leert met Grande Omega wel wat de uitkomst van een probleem kan zijn, zo'n functie. En daar blijft het wel bij, denk ik".

ST111 is iets positiever:

"*…voor een persoon die niet kan programmeren, ik vind het wel gewoon duidelijk …Als je geen ervaring hebt, wordt je wel gedwongen naar de practicumlessen te gaan en docenten hulp te vragen. Want...ik kom er niet doorheen. "*

ST38 is positief over de bijdrage van het leerprogramma aan het denken in logische volgordelijkheid:

"*…Grande Omega helpt om te oefenen op de logica...de logica van de functionaliteit, dat heb ik geleerd van Grande Omega."*

ST65 noemt proberen en logisch nadenken als aanpak:

"een beetje logisch nadenken, dan lukt het ook wel...als je ergens for loop voor nodig hebt, dat je dan gewoon getalletjes aanmaakt voor die for loop, zodat ie altijd blijft lopen…"

Over het maken van stapsgewijze specifieke instructies om een proces uit te voeren zeggen respectievelijk ST11 en ST32 het volgende:

"*…naarmate je gaat oefenen, dan ga je merken: oh, acht regels code die ik heb geschreven, kan ook in twee regels".*

"…dit doe ik nu anders…want als je het groot schrijft en het dan uittest, dan kan je heel precies je breakpoint zetten en weet je precies waar het fout gaat."

ST69 heeft van Grande Omega geleerd om het gedrag van de code op de achtergrond te begrijpen:

"Ik had vroeger nog wel eens dat ik hele dagen met een probleem zat, nu veel minder gelukkig."

ST23 was verrast dat Grande Omega een bijdrage leverde aan algoritmisch denken:

"I didn't expect it myself. I thought Grande Omega is not going to do anything…..But.. the whole idea was there in my mind".

Samengevat geven bijna alle studenten aan dat Grande Omega bijdraagt aan algoritmisch denken. De ervaren studenten gaan steeds efficiënter coderen. De beginners die met syntax en semantiek worstelen, vinden het moeilijk om de bijdrage van Grande Omega aan het denken in logische volgordelijkheid expliciet te maken. Ze merken wel dat ze meer logica gaan gebruiken. Uit tabel 5 blijkt dat het gemiddelde van de totaalscore bij het thema Algoritmisch denken 2.26 bedraagt op een schaal van 1-3.

3.8 Samenvatting invloed van het leerprogramma op CT

De deelvraag bij het kwalitatief onderzoek luidt: *Welke invloed heeft het gebruik van de leeromgeving Grande Omega op de ontwikkeling van Computational Thinking Skills?* De invloed van Grande Omega op de drie CT-thema's laat zich volgens de scores in tabellen 3, 4 en 5 het meest gelden bij het thema Algoritmisch denken (M=2.26). Bijna alle studenten geven aan dat Grande Omega bijdraagt aan algoritmisch denken. De ervaren studenten gaan steeds efficiënter coderen. De beginners die met syntax en semantiek worstelen, vinden het moeilijk om de bijdrage van Grande Omega aan het denken in logische volgordelijkheid expliciet te maken. Ze merken wel dat ze meer logica gaan gebruiken.

De invloed van Grande Omega op het thema Abstraheren (M=2.08) is 0.18 punten kleiner dan de score bij het thema Algoritmisch denken. Zeven van de negen studenten vinden dat Grande Omega nauwelijks tot geen bijdrage levert aan het onderscheiden of schematiseren van informatie. De studenten die gebruik maken van schematische hulpmiddelen, hadden al programmeerervaring voor zij aan de studie begonnen.

Bij het thema Denken in stappen (M=1.92) is de invloed het kleinst. De invloed is 0.34 punten kleiner dan bij het thema Algoritmisch denken. Het denken in stappen lijkt vooral voorbehouden aan het oplossen van programmeerproblemen. Ongeveer de helft van de negen studenten herkent de voordelen van een stapsgewijze aanpak bij het programmeren. Dit betekent niet dat deze studenten bij de aanpak van een project ook decompositie van een probleem toepassen. De meeste beginners die de theorie in Grande Omega hebben geleerd, merken dat ze nog onvoldoende vaardigheden hebben om werkende code te kunnen schrijven.

Tabel 6

Overzicht van uitspraken van geïnterviewde studenten, gerubriceerd per thema en per niveau. ST(00) verwijst naar de studentcode

Thema CT	Low level	Medium level	High level
Denken in stappen Een probleem opdelen in kleinere deelproblemen of in deelvragen. Een probleem zo formuleren dat het met behulp van een computer is op te lossen.	*"…Vaak kregen mensen gewoon taken, van hee, kan jij dit voor ons doen? Dan werd het aangeleverd, vervolgens als een persoon dat had aangeleverd, dan werd het in de groep gegooid. En dan ging iedereen kijken of hij nog zelf kon uitbreiden of verbeteren. Dus zelf code toevoegen aan het gemaakte deel…dat geleverd werd." (ST108)*	*"…Die tussenstap, daar waren we te laat achter gekomen. En daarna hebben we vanaf het herkansingmoment de draad weer opgepakt. Want we hadden de database al opgezet en vervolgens gingen we toen echt tutorials zoeken, een connectie maken met de database. En vervolgens dat je de gegevens uit de database filtert om het in een grafiek weer te geven" (ST111)*	*"…Dat zijn alle stappen en moet ik dus stap voor stap gaan kijken hoe ik dat moet doen. En dan in code om gaan zetten. Of opdelen van problemen. Als je een groot probleem hebt, krijg je het nooit opgelost". (ST69)*
Abstraheren De essentie verduidelijken zonder zich in details te verliezen. Schematiseren/model leren door gebruik te maken van schetsen, tabellen, grafieken of modellen.	*"…De belangrijkste onderdelen, op de manier van hoe we het uiteindelijk gedaan hebben? Oké. Voornamelijk de voorkennis die mensen al hadden in de projectgroep en Internet."(ST52)*	*"…De code is in die zin belangrijk, dat je iets moet hebben om te kunnen laten zien. Maar het hoeft niet ingewikkeld te zijn, als je het maar mooi kan presenteren. Dus die code is wel degelijk belangrijk, anders heb je niks. Maar buiten dat, als je het af hebt dan zorg je ervoor dat de randzaken ook allemaal in orde zijn". (ST68)*	*"…Uiteindelijk kwam ik dus met een tabelletje met alle requirements. In die requirements stonden bepaalde termen, termen die dus niet uitgelegd waren…Die moeten goed gedefinieerd worden voordat we kunnen beginnen aan het project en dat is dus ook wat we gedaan hebben." (ST32)*
Algoritmisch denken Stap-voor-stap specifieke en expliciete instructies maken om een proces uit te voeren. Logische volgordelijkheid toepassen.	*"…Nee, ja, maar dat was gewoon tussendoor. Zo van: hee, let je daar nog effe op? Maar dat werd niet behandeld in zo'n bestand, bijvoorbeeld op OneDrive. Dat was meer hoe ben je, doe je goed mee. Maar we gingen niet echt inhoudelijk in op de code …"(ST68)*	*"…Gewoon zoveel mogelijk opties, gekke dingen die je maar kan bedenken, altijd proberen. En zodra ik het uit zou willen geven wat ik heb gemaakt, dan ga ik gewoon aan kennissen en vrienden vragen van joh, kan je het testen? Om de een of andere reden krijgen ze het altijd voor elkaar om het alsnog te breken!" (ST52)*	*"…Door overal comments neer te zetten. En als het eenmaal werkt, dus ook wat netter te maken, dus dingen in functies zetten. Het zijn vaak heel lange codes en het wordt echt spaghetticode als we ook nog code to gaan gebruiken." (ST69)*

4. Conclusie en discussie

De hoofdvraag van dit onderzoek luidt: *Welke invloed heeft de online leeromgeving Grande Omega op de programmeerprestaties en de ontwikkeling van Computational Thinking Skills van eerstejaars studenten informatica bij Hogeschool Rotterdam?* In deze sectie worden de conclusies van beide onderzoeksdesigns weergegeven, gevolgd door de discussie en een beschouwing over de maatschappelijke relevantie van het onderzoek.

4.1 Conclusie kwantitatief onderzoek

Om het eerste deel van de hoofdvraag te beantwoorden - over de invloed van Grande Omega op de programmeerprestaties - staan vier hypotheses centraal.

De eerste hypothese: *Hoog presterende studenten (tentamencijfer ≥ 5,5) maken meer oefeningen en besteden meer tijd aan de oefeningen dan laag presterende studenten (tentamencijfer < 5,5),* wordt aangenomen. Uit de resultaten blijkt dat AA studenten significant meer oefeningen in Grande Omega maken dan BA studenten. AA studenten besteden eveneens significant meer tijd aan de oefeningen dan BA studenten.

De tweede hypothese: *Er is positieve samenhang tussen het aantal gemaakt oefeningen in Grande Omega, de tijd die aan de oefeningen is besteed en de prestaties, uitgedrukt in tentamenresultaten,* wordt aangenomen. Uit de resultaten blijkt dat er significante positieve samenhang is tussen het aantal gemaakte oefeningen in Grande Omega, de tijd die aan de oefeningen is besteed en de prestaties, uitgedrukt in tentamenresultaten.

De derde hypothese: *Studenten die veel fouten maken in Grande Omega presteren beter dan studenten die weinig fouten maken,* wordt aangenomen. Uit de resultaten blijkt dat AA studenten significant zowel meer correcte als incorrecte antwoorden bij de oefeningen geven dan BA studenten. Studenten die veel fouten maken in Grande Omega presteren dus beter dan studenten die weinig fouten maken.

De vierde hypothese: *AA studenten doen in elke onderwijsperiode meer oefensessies in Grande Omega dan BA studenten,* wordt deels aangenomen. In onderwijsperioden 1,3 en 4 oefenen de AA studenten significant meer dan de BA studenten. In onderwijsperiode 2 is het verschil niet significant. Om een verklaring te vinden voor de toename van het aantal oefensessies voor beide groepen in onderwijsperiode 3, is de analyse besproken met een DEV-docent. De docent verklaart dat de moeilijkheidsgraad van de oefeningen in onderwijsperiode 3 sterk toeneemt. Er worden nieuwe concepten in Grande Omega geïntroduceerd, zoals de programmeertaal C#. Dat zou de reden kunnen zijn dat beide groepen meer oefeningen gaan maken.

Tabel 7

Uitspraken van de geïnterviewde studenten over Grande Omega (GO)

* Studenten geven dit subjectieve cijfer aan hun eigen voortgang

Code en label interv.	**Vooropleiding en ervaring in programmeren**	**Programmeervaardigheden**			**Kernpunten waardering**	**Kernpunten kritiek**	**Wensen**
		*Vooruitgang in cijfer 1 t/m 10**	*Bijdrage GO*	*Naast GO geleerd van…*			
ST23 AA	Bachelor Engels. Geen IT ervaring.	0 naar 4 - 5.	100%	Geen.	Heeft mentaal model opgebouwd.	Ziet geen relatie met project.	Geen.
ST69 AA	Mbo ICT-beheer. Heeft IT ervaring.	4 naar 7.	50%	Zelfstandig onderzoek en projecten.	Leert van classes, wat code op de achtergrond doet.	Ziet geen relatie met project.	Heeft behoefte aan inhoudelijke feedback.
ST52 BA	Mbo Scheepsbouw. Heeft > 6 jaar IT ervaring.	5 naar 5,5.	0%	Klasgenoten, peer coaches en Internet.	Ziet perspectief in ontwikkeling van GO.	Taal van GO sluit niet aan op C#. Vindt dat leren in GO bestaat uit het oplossen van code van GO. Ziet geen relatie met project.	Wil leren door opdrachten te maken en daar feedback op te krijgen. Behoefte aan vrijheid programmeren.
ST32 BA	Havo met keuzevak Informatica. Beperkte IT ervaring.	4 naar 7.	30-40%	Hoorcolleges DEV.	Ziet GO als “poortwachter” om te controleren of code is begrepen.	Ziet geen relatie tussen GO en project, wel tussen GO en Analyse. Leert van GO de code van een ander te begrijpen, niet om zelf code te schrijven.	Heeft behoefte aan inhoudelijke feedback.
ST38 AA	Havo en deels WO Scheikunde. Geen IT ervaring.	1 naar 4.	65%	Projecten en YouTube.	Heeft in GO vooral de logica van de functionaliteiten geleerd.	Ziet groot verschil tussen wat geleerd wordt in GO en wat in het project moet worden toegepast.	Adviseert syntax van GO te vervangen door C#.
ST65 BA	Havo met keuzevak Informatica. Heeft IT ervaring.	7 naar 8,4.	15%	Project en zelfstudie	Leert van GO wat de uitkomst van een codeprobleem kan zijn.	Ziet geen relatie met project	Heeft behoefte aan debugger met inhoudelijke feedback.
ST68 A	Mbo Engineering. Geen IT ervaring.	0 naar 2.	60%	YouTube, klasgenoten en copy-paste.	Herkent GO niet als leermiddel om te coderen maar noemt wel leeropbrengsten.	Sprong naar OP3 was groot. GO hielp daar niet bij. Ziet geen relatie tussen GO en project of probleemoplossen.	Spreekt behoefte uit voor feedback tool.
ST111 BA	Havo en deels Bedrijfseconomie Geen IT ervaring.	Blijft 5 – 6.	70%	Practicum DEV	Leert van GO de basisprincipes van coderen. Noemt het maken van classes.	Ziet geen relatie met project.	Vindt dat practicumlessen noodzakelijk zijn. Behoefte aan regel voor regel nalopen van code, geleerde toepassen.
ST108 A	Mbo ICT-beheer. Heeft IT ervaring.	6-7 naar 8.	20%	Projecten.	Te theoretisch. Is positief over concept. Verwachtte dat GO getest zou zijn.	Ziet geen relatie met project. Vindt GO vooral theorie, niet zelf programmeren.	Heeft behoefte aan praktijk in programmeren.

Alle studenten maken een inhaalslag in de derde periode. De BA studenten oefenen dan niet alleen op de nieuwe, moeilijkere opgaven, maar halen ook de oude oefensessies op. In onderwijsperiode 4 oefenen alle studenten weer minder dan in onderwijsperiode 3.Een verklaring hiervoor zou kunnen zijn dat de meesten met de nieuwe concepten vertrouwd zijn geraakt. Dat de BA studenten ook minder oefeningen maken is opvallend. Juist zij voelen de druk oplopen, aangezien ze in het eerste jaar van de studie een minimaal aantal studiepunten moeten behalen om de studie te mogen vervolgen. De docent denkt dat de BA-studenten onvoldoende resultaat hebben bereikt en het "opgeven" in onderwijsperiode 4. De golfbeweging die in figuur 2 zichtbaar is, vraagt om nader onderzoek, bijvoorbeeld naar motivationele aspecten, de opbouw van complexiteit van de oefeningen en de aard van de online feedback. Zodat op basis van deze kennis interventies kunnen worden gedaan die de leeropbrengsten positief kunnen beïnvloeden.

De invloed van Grande Omega op de programmeerprestaties blijkt afhankelijk van de mate van inspanning en tijdsbesteding van studenten bij het leerprogramma (Robins et al., 2003). Hoog presterende studenten leren van zowel correcte als incorrecte antwoorden bij het maken van de oefeningen. Deze bevinding sluit aan op onderzoeken van Mathan en Koedinger (2005) en Grassinger et al. (2018).

4.2 Conclusie kwalitatief onderzoek

Om het tweede deel van de hoofdvraag te beantwoorden - over de invloed van Grande Omega op de ontwikkeling van Computational Thinking Skills - staan de drie CT-thema's centraal, voorafgegaan door een korte beschouwing van de algemene aspecten van Grande Omega.

De meeste studenten staan welwillend tegenover de ontwikkeling van het leerprogramma. Wel zijn ze kritisch over de gebruikte syntax, de betrouwbaarheid van het tentamenprogramma en de wijze van feedback. Over de aansluiting van Grande Omega op het projectonderwijs oordelen ze negatief, specifiek wanneer ze hun verworven programmeervaardigheden willen toepassen. Qian en Lehman (2017) en Robins et al. (2003) bevestigden eerder dat beginners het moeilijk vinden om programmeerkennis toe te passen.

Ten aanzien van de ontwikkeling van hun Computational Thinking Skills zijn de oordelen van de geïnterviewden bij de thema's Denken in stappen, Abstraheren en Algoritmisch denken als volgt: De invloed van Grande Omega laat zich het meest gelden bij het thema Algoritmisch denken. Bijna alle studenten geven aan dat Grande Omega hieraan bijdraagt. De ervaren studenten gaan steeds efficiënter coderen. De beginners die met syntax en semantiek worstelen, vinden het moeilijk om de bijdrage van Grande Omega aan het denken in logische volgordelijkheid expliciet te maken. Ze merken wel dat ze meer logica gaan gebruiken. De invloed van Grande Omega op het thema Abstraheren is kleiner dan bij het thema Algoritmisch denken. Zeven van de negen studenten vinden

dat Grande Omega nauwelijks tot geen bijdrage levert aan het onderscheiden of schematiseren van informatie. De studenten die gebruik maken van schematische hulpmiddelen, hadden al programmeerervaring voor zij aan de studie begonnen.

Bij het thema Denken in stappen is de invloed het kleinst. Het denken in stappen lijkt vooral voorbehouden aan het oplossen van programmeerproblemen. Ongeveer de helft van de negen studenten herkent de voordelen van een stapsgewijze aanpak bij het programmeren. Dit betekent niet dat deze studenten bij de aanpak van een project ook decompositie van een probleem toepassen. De meeste beginners die de theorie in Grande Omega hebben geleerd, merken dat ze nog onvoldoende vaardigheden hebben om werkende code te kunnen schrijven.

In de reviewstudie van Voogt et al. (2017), zijn, met enig voorbehoud, positieve effecten van programmeeronderwijs op CT in het basis- en voortgezet onderwijs gevonden. In relatie tot dat onderzoek lijkt Grande Omega positieve invloed te hebben op de ontwikkeling van computationele concepten en computationele methoden bij hbo-studenten informatica, maar niet op generieke kennis en -vaardigheden, zoals probleemoplossend vermogen buiten een programmeeromgeving.

4.3 Discussie

4.3.1 Kwantitatief onderzoek

Het onderzoeksontwerp kent enkele beperkingen. Grande Omega staat als leermiddel niet volledig op zichzelf. De practicumlessen vormen een belangrijke aanvulling. Het DEV-programma heeft bovendien een inhoudelijke relatie met andere leerlijnen. Enkele voorbeelden van variabelen die van invloed kunnen zijn op de mate van inspanning en de resultaten van studenten in Grande Omega: a) de pedagogisch-didactische vaardigheden en de vakinhoudelijke expertise van DEV-docenten bij de practicumlessen, b) de onderwerpen die bij de leerlijn Analyse aan bod komen en c) de demografische en sociaaleconomische kenmerken van studenten en d) de mate van begeleiding van studieloopbaancoaches. Uit het vooronderzoek en het kwalitatief onderzoek blijkt bovendien dat studenten inventief zijn bij het behalen van hun leerdoelen. Ze leren, naast Grande Omega, ook van YouTube, van gesprekken met peer coaches en van klasgenoten.

Grande Omega is doorlopend in ontwikkeling, zowel technisch als inhoudelijk. Aangezien docenten en studenten per kwartaal feedback geven aan de makers, zullen de aard, de vorm en de frequentie van de oefensessies via iteraties veranderen.

Toekomstig onderzoek zou zich kunnen richten op het verband tussen het type opdrachten en de leeropbrengsten van studenten. Welke opdrachten worden als zeer moeilijk ervaren en waarom? Het kan ook waardevol zijn om te onderzoeken in welke mate betrokkenheid van peer coaches en klassenvertegenwoordigers bij de ontwikkelingsfases van het leerprogramma tot hogere leeropbrengsten bij de studenten leidt. Hun inzichten in de leercurve van de studenten kan aanvullende

informatie opleveren. Onderzoek dat zich richt op motivationele (Ryan & Deci, 2000) en onderwijskundige factoren, zoals scaffolding (Hoogveld, Janssen-Noordam & Van Merriënboer, 2011), kan verder richting geven aan de didactische opbouw van het programma. Door de relaties tussen vooropleiding, voorkennis en resultaten te onderzoeken (Dick, Carey & Carey, 2009) is input voor een gedifferentieerde opzet van het programma te verwerven, waardoor meer studenten zich thuis voelen bij het leerprogramma.

4.3.2 Kwalitatief onderzoek

Het kwalitatief onderzoek kent eveneens een aantal beperkingen. Ten eerste zijn de interpretaties gebaseerd op de meningen van negen select gekozen studenten. Ten tweede zijn de drie thema's onderzocht aan de hand van hun ervaringen bij één project. Ten derde interpreteerden studenten de vragen niet altijd hetzelfde. Vooral bij het thema Abstraheren bleek het lastig om een bevredigend beeld te krijgen van de onderliggende CT-vaardigheden. Het type vragen was misschien minder geschikt om dit aspect te verhelderen. Het was wellicht beter geweest om de hele set aan vragen ook met enkele eerstejaars studenten te testen. Dit heeft echter twee nadelen: a) het vraagt om aanzienlijk meer tijd en b) er is een risico op verspreiding van de ingreep, met bias als gevolg, omdat studenten de vragen onderling met elkaar kunnen bespreken en tijdens het interview gewenste antwoorden zouden kunnen geven. Ten vierde is het onderzoek gelimiteerd tot de informaticaopleiding van Hogeschool Rotterdam, aangezien het leerprogramma vooralsnog alleen daar is ingezet.

De bijdrage van Grande Omega aan het ontwikkelen van Computational Thinking Skills kan niet worden losgezien van de context van het gehele onderwijsprogramma. In het vooronderzoek gaven peer coaches en docenten aan dat, ten aanzien van het thema Denken in stappen en Abstraheren, eerstejaarsstudenten waarschijnlijk meer leren van de leerlijn Analyse dan van het programma Grande Omega binnen de leerlijn DEV. Uit het onderzoek blijkt dat de bijdrage van Grande Omega aan Abstraheren gemiddeld is en dat de bijdrage aan Denken in stappen klein is. De leerlijn Analyse wordt in dit verband echter slechts enkele keren door de geïnterviewde studenten genoemd. Studenten kunnen niet altijd expliciet benoemen waar zij hun leeropbrengsten aan te danken hebben. Als ze zeggen dat ze vooral van hun klasgenoten of de peercoaches leren, kan het zijn dat ze indirect de leerstof van een cursus eigen maken. Het kan ook zijn dat ze vooral leren van Internet, zoals velen aangeven. Voor toekomstig onderzoek is het daarom waardevol om te analyseren in welke mate deze groep baat heeft bij informeel leren versus formeel leren (Cox, 2012).

Conform de meningen van de peer coaches en docenten in het vooronderzoek draagt Grande Omega bij aan Algoritmisch denken. Uit het onderzoek blijkt dat dit wordt bevestigd door de studenten, met de aantekening dat Algoritmisch denken vooral binnen het programmeeronderwijs plaatsvindt. Er zijn geen aanwijzingen gevonden dat studenten deze denkwijze bij het projectonderwijs toepassen. In dit

onderzoek is geen bewijs gevonden dat Grande Omega de algemene probleemoplossende vaardigheden van studenten verbetert. Hier zou een relatie kunnen zijn met de opmerkingen van studenten over de vorm en aard van de feedback. Veel studenten geven aan dat ze meer inhoudelijke feedback wensen. De uitgestelde feedback van het systeem stimuleert hen onvoldoende om te reflecteren op het onjuiste antwoord, omdat ze niet weten wat ze goed of fout hebben gedaan en hoe ze zich de leerstof eigen kunnen maken. Inhoudelijke feedback die just-in-time wordt aangeboden en studenten uitdaagt tot deep learning (Kleij, Feskens & Eggen, 2015, Schneider & Preckel, 2016), stimuleert reflectie en probleemoplossende vaardigheden. Studenten spreken hun voorkeur uit voor meer praktijkgerichte oefeningen die aansluiten bij de gevraagde programmeervaardigheden bij het projectonderwijs. Dit bevordert de transfer van kennis en vaardigheden (Ambrose et al., 2010). Deze praktijkgerichte oefeningen kunnen de Computational Thinking Skills bevorderen (Fisser & Strijker, 2016), omdat ze op meer CT-vaardigheden dan alleen Algoritmisch denken een beroep doen.

Toekomstig onderzoek zou zich kunnen richten op het omgekeerde van de benadering die hier is gekozen: welke bijdrage levert het specifiek trainen van Computational Thinking Skills bij andere cursussen aan het programmeeronderwijs? Een vervolgvraag kan zijn: zouden door deze benadering informaticastudenten beter toegerust zijn om met minder inspanning een mentaal model te ontwikkelen?

4.4 Maatschappelijke relevantie

Hbo-studenten Informatica worden opgeleid voor een IT-positie in het bedrijfsleven. Vertegenwoordigers uit het beroepenveld benadrukken dat de meeste informaticabedrijven als lerende organisaties zijn ingericht, waarbij het samenwerken en kennis delen als kerncompetenties gelden. Vanuit epistemologisch en ontologisch perspectief is dit een constructivistische benadering (Valcke, 2009). Grande Omega is echter gebaseerd op een cognivistische visie (Ertmer & Newby, 1993): de instructie is sterk geordend en gestructureerd. De oefeningen doen een beroep op declaratieve, procedurele en metacognitieve kennis. Het accent ligt bij Grande Omega op cognitief leren. Vormen van spontaan leren blijven op de achtergrond. Er is weinig aandacht voor persoonlijke ervaringen en sociale interactie tussen studenten onderling. Bij de practicumbijeenkomsten heeft de instructieverantwoordelijke een leidende en sterk sturende rol (Valcke, 2009).

De geïnterviewde studenten spreken impliciet hun voorkeur uit voor een meer sociaal-constructivistische benadering, zoals uit tabel 7 is op te maken: kennisconstructie die in relatie met anderen plaatsvindt (Duffy & Cunningham, 1996). Ze erkennen de bijdrage van Grande Omega aan hun leerproces maar geven tegelijkertijd signalen af dat ze behoefte hebben aan het delen van betekenissen en het leren door kennis toe te passen. Ze wensen gezamenlijk kennis op te doen die

afhankelijk is van de context waarin deze is verworven. Vaak is dit het toepassen van het geleerde via authentieke problemen in een praktijkgerichte setting (Valcke, 2009).
Bij hogeschool Rotterdam is het macroperspectief op didactisch handelen uitgewerkt in aanbevelingen voor een constructivistische benadering. Op mesoniveau kan vanuit richtinggevende beleidskaders invloed op de instructievormen worden uitgeoefend. Op microniveau is het mogelijk om meer constructivistische elementen toe te voegen zoals a) een open leeromgeving waarin studenten actief kennis kunnen delen, b) aanpassingen van het probleemgestuurd onderwijs door de leerstof van DEV bij projecten in een grotere context te plaatsen en c) de rol van de lesgevende docent naar begeleider en coach om te buigen. De docenten kunnen dan meer aan de behoefte voor feed up, feed back en feed forward (Ambrose et al., 2010, Valcke, 2009) tegemoet komen. Wanneer de keuzes bij het didactisch handelen gestuurd worden door de leerbehoeften van studenten, in combinatie met de wetenschappelijke bevindingen over leren programmeren, sluit Grande Omega optimaal aan bij de 21ste eeuwse vaardigheden die in de beroepspraktijk worden beoefend.

Referenties

Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M.C., & Norman, M. K. (2010). *How learning works, seven research-based principles for smart teaching.* San Francisco, USA: Jossey-Bass.

Anderson, J. R., Conrad, F. G. & Corbett, A. T. (1989). Skill Acquisition and the LISP Tutor. *Cognitive Science, 13,* 467-505.

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behaviour change. *Psychological Review, 84*(2), 191-215.

Bandura, A. (1993). Perceived self-efficacy in cognitive development and functioning. *Educational Psychologist, 28,* 117–148.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2*(1), 48-54.

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper for 2012 annual meeting of the American Educational Research Association. Vancouver: Canada.

Cox, M.J. (2012). Formal to informal learning with IT: research challenges and issues for e-learning. *Journal of Computer assisted Learning, 29 (1),* 85-105.

CSTA & ISTE (2009). Computer Science Teachers Association & International Society for Technology in Education. Op 3 maart, 2018, verkregen van: https://ec.europa.eu/jrc/en/computational-thinking

Czerkawski, B.C. & Lyman, E.W. (2015). Exploring Issues About Computational Thinking in Higher Education, *TechTrends, 59*(2) 57-65.

Dick, W., Carey, L., & Carey, J. O. (2009). *The systematic design of instruction* (7th ed.). Upper Saddle River: Pearson.

Duffy, T. M., & Cunningham, D. J. (1996). *Constructivism: Implications of the design and delivery of instruction*. In D. Jonassen (Ed.), Handbook of Research for Educational Communications and Technology, 170-195. London: Prentice Hall.

Driscoll, M. P. (2005). *Psychology of learning for instruction.* Derde druk. USA: Pearson Education.

Ertmer, P., & Newby, T. (1993). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance Improvement Quarterly, 6 (4),* 50-72.

Fisser. P., & Strijker, A. (2016). *Computational thinking in het kader van 21e eeuwse vaardigheden*. Paper gepresenteerd op de VELON-conferentie, Brussel.

Grassinger R., Scheunpflug A., Zeinz. H. & Dresel, M. (2018). Smart is who makes lots of errors? The relevance of adaptive reactions to errors and a positive error climate for academic achievement. *High Ability Studies, 29* (1), 37-49. doi: 10.1080/13598139.2018.1459294

Hattie, J. (2009). *Visible Learning: A synthesis of over 800 meta analyses relating to achievement*. London: Routledge.

Hattie J. & Timperly, H. (2007). The power of feedback. *Review of Educational Research, 77*(1), 81-112.

Hoogveld, B., Janssen-Noordam, J., & Merriënboer. J. J. G. van. (2011). Innovatief onderwijs in de praktijk: Toepassingen van het 4C-ID-model. Groningen: Noordhoff.

Kleij van der, F. M., Feskens, C. W. & Eggen, J. H. M. (2015). Effects of feedback in a computer-based learning environment on students' learning outcomes: a meta-analysis. *Review of Educational Research, 85,* (4), 475-511.

Komarraju, M. & Nadler, D. (2013). Self-efficacy and academic achievement: Why do implicit beliefs, goals, and effort regulation matter? *Learning and Individual Differences, 25,* 67-72.

Lahtinen, E., Ala-Mutka, K. & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin, 37,* (3).

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Mathan, S. A. & Koedinger, K. R. (2005). Fostering the Intelligent Novice: Learning from errors with metacognitive tutoring. *Educational Psychologist, 40*(4), 257-265.

Milne, I. & Rowe, G. (2002). Difficulties in learning and teaching programming - views of students and tutors. *Education and Information Technologies, 7*(1), 55-66.

Mory, E. H. (2004). Feedback research revisited. In D. H. Jonassen (Ed.), *Handbook of research for educational communications and technology.* New York: Simon & Schuster Macmillan.

Pintrich, P. R. (2004). A conceptual framework for assessing motivation and selfregulated learning in college students. *Educational Psychology Review, 16,* 385-407.

Qian, Y. & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: a literature review. *ACM Transactions on Computer Education, 18* (1). doi: https://doi.org/10.1145/3077618

Robins, A., Rountree, J. & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2),137-172.

Ryan, R. M. & Deci, E. L., (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology 25,* 54-67.Verkregen op 3 maart, 2017, van http://www.idealibrary.com.

Schneider, M. & Preckel, F. (2016). *Variables Associated with Achievement in Higher Education: A Systematic Review of Meta-analyses.* Manuscript. Accepted for publication at Psychological Bulletin, December 20, 2016.

Valcke, M. (2009). Onderwijskunde als ontwerpwetenschap. Gent, België: Academia Press.

Voogt, J., Brand-Gruwel, S. & Van Strien, J. (2017). *Effecten van programmeeronderwijs op computational thinking.* Reviewstudie. Verkregen op 1 september, 2017, van: https://www.nro.nl/wpcontent/uploads/2017/05/003-Antwoord.

Wing, J. (2006) Computational thinking. *Communication in ACM, 49*(3), 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A, 36*6(1881), 3717- 3725.

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research, 53*(4), 562-590. doi: http://dx.doi.org/10.1177/0735633115608444.

Zimmerman, B. J. (Ed.). (1990). Self-regulated learning and academic achievement. *Educational Psychologist, 25*(1).

Bijlagen

Bijlage 1. Vooronderzoek CT

Tijdstip van het onderzoek

Het onderzoek vindt plaats tussen 22 april 2018 en 28 juli 2018 in onderwijsperiode 4 van de opleiding Informatica bij hogeschool Rotterdam.

Vooronderzoek CT

Bij de kwalitatieve analyse onderzoekt de thesisstudent op welke wijze en in welke mate de ondernomen leeractiviteiten in GO bijdragen aan de ontwikkeling van Computational Thinking Skills (CT). Dit gebeurt door open interviews af te nemen met een tiental studenten uit leerjaar 1, waarbij de bijdrage van GO aan hun persoonlijke ontwikkeling van CT centraal staat. CT staat als één van de 21st Century Skills voor een set van probleemoplossende vaardigheden en houdingen, waarbinnen een student gebruikmaakt van concepten uit de informatica. Deze concepten hebben vooral betrekking op a) het kunnen abstraheren, b) het denken in stappen, c) het analytisch en algoritmisch denken, d) het ordenen van informatie e) het besef van volgordelijkheid en f) het modelleren van gegevens. De nadruk ligt bij het onderzoek op de drie concepten abstraheren, het denken in stappen en algoritmisch denken. Docenten en studenten geven in het vooronderzoek aan dat deze aspecten voor informatici de belangrijkste cognitieve bouwstenen vormen. Tabel 1 biedt een overzicht van de informatie die uit de verkennende gesprekken met peer-coaches en docenten is verkregen. Peer coaches zijn studenten uit het tweede jaar van de opleiding die de eerstejaars ondersteunen bij het leren programmeren.

Tabel 1

Overzicht van de informatie die uit de verkennende gesprekken met peer-coaches en docenten is verkregen. De afkorting GO staat voor Grande Omega

Samenvatting meningen 4 peercoaches		**In welke mate draagt GO bij aan…**		
Voordelen GO	**Nadelen GO**	**Abstraheren?**	**Stapdenken?**	**Algoritmisch?**
Je leert juiste code in goede volgorde zetten. Het klopt allemaal en het oefenen met GO is handig en nuttig voor beginnende informatici. Het is theoretisch. Het is een goed oefenplatform met een opbouw in moeilijkheid, de instructie is minder	Het is vooral coderen, niet echt programmeren. Je krijgt geen feedback, alleen een rood blokje. Je weet dus niet waarom iets goed of fout is. Sommige studenten leren antwoorden uit het hoofd. De teksten in	In combinatie met Analyse wel. De stap bijvoorbeeld van code naar UML is kleiner. Bij DEV is die relatie minder duidelijk.	Decompositie van een probleem: nee, GO draagt hier niet aan bij. Projecten 2 en 3 wel.	Je leert denken in algoritmen. De opbouw van de vragen in GO ondersteunt dit.

goed. De invloed van een docent maakt veel uit.	de readers zijn te ingewikkeld voor eerstejaars. Voor sommige studenten gaat de opbouw in GO te snel.			
Mening docent 1		**In welke mate draagt GO bij aan…**		
Voordelen GO	**Nadelen GO**	**Abstraheren?**	**Stapdenken?**	**Algoritmisch?**
Het is een sterke tool omdat je leert wat er op de achtergrond speelt wanneer je gaat programmeren. Zodat je minder fouten maakt, omdat je leert denken als een computer.	Je oefent te weinig in het schrijven van kleinere programma's, terwijl je dat wel moet doen in de projecten. Er zit een gat tussen wat je in GO oefent en wat je moet toepassen in de projecten.	In project 3 moesten studenten functies en klassen gebruiken om te programmeren. Daarvoor moesten ze abstract denken. Analyse hielp daarbij.	Sterk punt: wat in het geheugen van GO zit, staat ernaast dus je kunt zien wat al gedaan is. Zo leer je in stappen denken, net als de computer.	Ik zie algoritmisch als methodisch, als een structuur aanbrengen. In die zin draagt GO bij aan gestructureerd denken.
Mening docent 2		**In welke mate draagt GO bij aan…**		
Voordelen GO	**Nadelen GO**	**Abstraheren?**	**Stapdenken?**	**Algoritmisch?**
GO helpt bij het nauwkeuring evalueren van het gedrag van een computer. Je leert regels over hoe je het algoritme in machine-instructies kunt vertalen.	De toepassing van wat je leert ontbreekt. Je leert dus niet om kleine programma's te schrijven. Metafoor: je leert ontbrekende woorden in zinnen op te sporen, maar je leert niet om met die kennis een alinea te schrijven.	GO draagt niet direct bij aan de vaardigheid om in abstracties te denken of te schematiseren of modelleren.	GO helpt studenten wel om in stappen te denken. Problem solving is echter geen doel van GO.	GO helpt bij het leren opsporen van fouten, zodat je de oorzaak van bugs kan vinden.
Mening docent 3		**In welke mate draagt GO bij aan…**		
Voordelen GO	**Nadelen GO**	**Abstraheren?**	**Stapdenken?**	**Algoritmisch?**
Ze leren coderen. Ze zijn beter voorbereid op het leren van verschillende programmeertalen. GO legt hiervoor een basis. Studenten met voorkennis (programmeerkennis) vinden het leuk.	De theorie komt te vroeg. Beginners hebben nog geen kapstok. Het zijn invuloefeningen. Er is geen informatie over een programmeerprobleem waardoor ze zelfstandig een vertaalslag kunnen maken.	Leren ze vooral bij Analyse: DFD, Lifecycle, de relevante kern eruit halen, het herformuleren en meetbaar maken.	Ze leren niet denken in problemen. GO gaat over theoretische concepten. Stapsgewijs van een draft naar begrijpelijk, (niet ambigue) cyclisch denken gebeurt bij meer bij Analyse.	Ze leren coderen in GO zodat ze vertrouwd raken met algoritmes.

De docenten zijn software engineers die les geven in de leerlijnen Analyse, Development en Projecten. De thesisstudent heeft hen gevraagd hoe de drie concepten abstraheren, het denken in stappen en algoritmisch denken bij eerstejaars tot uitdrukking komen. Deze informatie, in combinatie met literatuur over het assessen van CT, wordt gebruikt om het open interview vorm te geven.

De drie concepten zijn als volgt gedefinieerd op basis van literatuuronderzoek over CT en verkennende gesprekken met docenten en studenten:

1. **Het denken in stappen** wordt decompositie genoemd. Het vermogen om een groot of complex probleem op te delen in kleinere deelproblemen en op die wijze naar een of meerdere oplossingsrichtingen toe te werken.
2. **Abstraheren** is een essentieel onderdeel va CT: het omvat het proces van het verduidelijken, modelleren en schematiseren van een complex onderwerp door onnodige details weg te laten.
3. **Algoritmisch denken** is het vermogen om stap-voor-stap specifieke en expliciete instructies voor het uitvoeren van een proces vorm te geven.

Tabel 2

Thema´s Computational Thinking Skills met indicatoren

Hoofdthema CT	**indicator**	**indicator**
Denken in stappen	Een probleem opdelen in kleinere deelproblemen of in deelvragen	Een probleem zo formuleren dat het met behulp van een computer is op te lossen
Abstraheren	De essentie verduidelijken zonder zich in details te verliezen	Schematiseren/modelleren door gebruik te maken van schetsen, tabellen grafieken, modellen (zoals context- of data-flow diagrams, class diagrams, simulaties, etc.)
Algoritmisch denken	stap-voor-stap specifieke en expliciete instructies maken om een proces uit te voeren	Logische volgordelijkheid toepassen

Iteratie 1

De concept-themavragen zijn in de eerste iteratie door twee afstudeerstudenten doorgenomen. Ze hebben aanbevelingen gedaan om de vragen begrijpelijker en concreter te maken voor de studenten uit het eerste jaar.

Iteratie 2

Alle vragen zijn vervolgens toegespitst op de praktijkervaringen van studenten in het eerste leerjaar en de semantiek die zij hanteren. De afstudeerstudenten en de peer coaches hielpen bij de tweede iteratie met het bedenken van mogelijke antwoorden op de vragen, om het herkennen van trefwoorden te bevorderen. Tenslotte gaven ze mee dat de meeste studenten uit leerjaar 1 modelleren (bv. UML) pas doorgronden als ze zelf hebben geprogrammeerd en hun eigen code begrijpen.

Iteratie 3

In de volgende iteratie zijn de aangepaste vragen doorgenomen met een peercoach. De peercoach schetste mogelijke antwoorden van studenten, zodat de vragen verder konden worden aangescherpt.

De lijst met trefwoorden is met behulp van de peercoach verder uitgebreid. De peercoach gaf mee dat studenten uit leerjaar 1 nog niet vertrouwd zijn met gangbare programmeertermen en soms termen verwisselen. De verwachting is dat ze in eigen woorden hun werkwijze proberen te omschrijven.

Tabel 3

Thema´s Computational Thinking Skills met mogelijke trefwoorden

Thema	**Mogelijke trefwoorden**
1. Denken in stappen	Praten met elkaar, (via Scrum) op hoogte van wat er gebeurt, opdelen in groepjes, subvragen verdelen, opdelen van het probleem, deelproblemen, volgorde: data ophalen, connectie database, opvragen data, terugkrijgen data, verwerken data, programmeren en visualiseren data.
2. Abstraheren	Hele dataset van het project bekijken, naar kern probleem kijken, relevant, schetsjes maken, schema, workflow, tabel, simulatie.
3. Algoritmisch denken	Variabelen, branching (if/else) loops, classes, interfaces, functies, subroutines, input, verwachtte output, logica, volgorde, testen.

Indicatoren niveau antwoorden bij themavragen

1. Vragen over Denken in Stappen

Een probleem opdelen in kleinere deelproblemen of in deelvragen.

Een probleem zo formuleren dat het met behulp van een computer is op te lossen.

Tabel 4

Thema Denken in stappen met indicatoren niveau

Denken in stappen	laag (L)	gemiddeld (M)	hoog (H)
Beschrijf hoe je project 3 vanaf het begin aanpakte.	Student geeft een rudimentaire beschrijving, maar geen details over de aanpak.	Student beschrijft een algemeen beeld van het project en noemt enkele losse componenten.	Student beschrijft gedetailleerd hoe hij/zij in stappen met onderdelen van het project is omgegaan.
Hoe hield je overzicht over wat je moest doen?	Student geeft geen specifieke voorbeelden	Student geeft een algemeen voorbeeld van een poging	Student beschrijft een specifieke, gestructureerde aanpak of planning
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en hoe je dat deed.	Student zegt dat er geen revisies of iteraties waren of geeft geen voorbeelden van revisies of iteraties	Student beschrijft een specifiek voorbeeld van een revisie of iteratie	Student beschrijft specifieke revisies of iteraties, vertelt het verloop en beargumenteert de keuzes
Hoe ben je met het omvangrijke probleem van project 3 omgegaan, zodat je het met code kon oplossen?	Student geeft geen beschrijving	Student geeft een voorbeeld van een stapsgewijze oplossing	Student beschrijft specifieke en of meerdere stapsgewijze oplossingsrichtingen
Op welke wijze helpt Grande Omega bij het oplossen van zo'n probleem?		Positieve en/of negatieve voorbeelden	

2. Vragen over Abstraheren

De essentie verduidelijken zonder zich in details te verliezen. Schematiseren/modelleren door gebruik te maken van schetsen, tabellen grafieken, modellen (zoals context- of data flow diagrams, class diagrams, simulaties, etc.)

Tabel 5

Thema Abstraheren met indicatoren niveau

Abstraheren	laag (L)	gemiddeld (M)	hoog (H)
Wat waren de belangrijkste onderdelen van project 3?	Student kan de essentiële informatie niet isoleren	Student beschrijft hoe hij de essentiële informatie voor het programmeerdeel isoleert	Student beschrijft een specifieke aanpak met schematische hulpmiddelen
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?	Student geeft geen voorbeelden waaruit blijkt dat hij onderscheid maakt tussen relevante en minder relevante kwesties	Student geeft een voorbeeld van een aanpak om relevante en minder relevante kwesties te scheiden	Student beschrijft een specifieke aanpak met niveaus van bijvoorbeeld requirements. Of noemt classes, tabellen, modellen diagrammen enz.
Op welke wijze helpt het oefenen in Grande Omega je hierbij?		positieve en/of negatieve voorbeelden	

3. Vragen over Algoritmisch denken

Stap-voor-stap specifieke en expliciete instructies maken om een proces uit te voeren. Logische volgordelijkheid toepassen.

Tabel 6

Thema Algoritmisch denken met indicatoren niveau

Algoritmisch denken	laag (L)	gemiddeld (M)	hoog (H)
Hoe zet jij een probleem om in code?	Student beschrijft geen strategie of een ineffectieve strategie	Student beschrijft een strategie, noemt enkele termen	Student benoemt specifieke stappen in logische volgorde, zet denkwijze uiteen
Hoe ga je na of jouw code correct functioneert? (Als je merkt dat jouw code niet goed werkt, hoe ga je dan te werk)?	Student beschrijft geen aanpak of een ineffectieve aanpak	Student beschrijft een algemene aanpak en noemt variabelen, input en output	Student beschrijft een specifieke situatie, een oplossing en hoe hij controleert of de oplossing werkt, bv via testen
Hoe zorg je ervoor dat jouw code werkbaar is voor anderen?	Student beschrijft geen kenmerken	Student noemt een aantal kenmerken waaraan de code moet voldoen	Student noemt een aantal kwaliteitskenmerken, noemt voorwaarden voor onderhouden of uitbreiden
Heb je wel eens	Student beschrijft geen	Student noemt een	Student geeft specifiek

teruggekeken naar je oude code? Wat zou je nu anders doen?	verbeterpunten	verbeterpunt of toekomstige aandachtpunten	aan wat hij/zij nu heeft begrepen, wat student voortaan anders zou doen of verwacht beter te doen
Op welke wijze helpt het oefenen in Grande Omega je hierbij?		Positieve en/of negatieve voorbeelden	

Bijlage 2. Interviewprotocol

Interviewprotocol onderzoek Grande Omega OP4 2018

Tijdstip:
Plaats:
Interviewer:
Studentnummer geïnterviewde:

Het studentnummer is na het verzamelen van de data gecodeerd zodat de antwoorden niet herleidbaar zijn naar de persoon.

- Omschrijving van het onderzoek
- Doel van het onderzoek
- Duur van het interview
- Protocol geluidopnamen en transcript
- Check van informed consent formulier
- Test audio recorder

Algemene vragen

- Wat is je vooropleiding?
- Omschrijf je programmeerervaring. Wat kon je al toen je begon aan dit collegejaar?
- Als je jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?
- Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden na dit collegejaar, welk cijfer is dat dan? Om welke reden?
- Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?

Themavragen DIS

- Beschrijf hoe je project 3 vanaf het begin aanpakte.
- Hoe hield je het overzicht over wat je moest doen?
- Beschrijf wat je bij je revisies of iteraties (bv. Scrum) deed en hoe je dat deed.

- Hoe ben je met het omvangrijke probleem van project 3 omgegaan, zodat je het met code kon oplossen?
- Op welke wijze helpt Grande Omega bij het oplossen van zo'n probleem?

Themavragen AB

- Wat waren de belangrijkste onderdelen van project 3?
- Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?
- Op welke wijze helpt het oefenen in Grande Omega hierbij?

Themavragen ALD

- Hoe zet jij een probleem om in code?
- Hoe ga je na of jouw code correct functioneert?
  (Als je merkt dat jouw code niet goed werkt, hoe ga je dan te werk?)
- Hoe zorg je ervoor dat jouw code ook werkbaar is voor anderen?
- Heb je wel eens teruggekeken naar je oude code? Wat doe je nu anders?
- Op welke wijze helpt het oefenen in Grande Omega hierbij?

Bijlage 3. Transcript interviews

Transcript interview 1 _onderzoek Grande Omega OP4 2018

Datum: maandag 4 juni 2018

Plaats: HR Rotterdam

Interviewer: AvB

Geïnterviewde: 1, studentcode 23

Algemene vragen

vraag	**antwoord**	*kenmerken*
Wat is je vooropleiding?	I have a bachelor of language of English. After that I had sort of job experiences, but they had nothing to do with programming.	Bachelor in buitenland.
Omschrijf je programmeer-ervaring. Wat kon je al toen je begon aan dit collegejaar?	Zero. I had no idea about programming before. Yeah, never - you know - encountered it in my life.	
Programmeerde je nooit een beetje voor je plezier?	Not at all.	Geen voorkennis en geen ervaring in programmeren.
Als je jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?	It would be zero.	
Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeer-vaardigheden na dit collegejaar, welk cijfer is dat dan? Om welke reden?	I think 4 or 5. But I am sort of a perfectionist, so I see myself way less than what it is. I really don't dare to say more than 4 or 5.	Van 0 naar 4 of 5.
En als je naar je cijfers van dit jaar kijkt, zijn ze bovengemiddeld?	All of them, yeah. I even had a 10 for programming, for development, DEV. And for Analyse I had 9,3. So it is all above 8. Without redoing the exams. It was the first time I had all of them. So far…	
Hoe komt het dat er zo'n verschil is tussen het gemiddelde cijfer wat je jezelf geeft en wat de opleiding je geeft?	Yeah. Because, I believe…that because I'm good at studying and I am good at solving problems, sort of like Grande Omega, a sort of puzzle. So I can…I am really good at doing puzzles I think. That's fine, it is easy to solve for me. If I look, it's like a puzzle…sort of mathematics…but not really as programming, maybe.	Houdt van probleem-oplossen en is goed in wiskunde.
Was je goed in wiskunde op school?	Yes, I was! Like twenty years ago. I don't remember anymore, or more (laughs).	
Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?	Yeah, mostly it's related to Grande Omega, because the most important part of study is programming. Yeah, next to it we have Analyse but that is not always directly related to programming. So my programming talent, it's all coming from Grande Omega.	100% GO
Wat heb je geleerd?	With Grande Omega you mean, specifically? All the basics. That, I really find very good. And we are going to construct you know, these basics as a blueprint. Just learn	Heeft alle basisvaardig-

	the fundamental things. How, just the functionalities, how it works.	heden van GO geleerd.
Als je zegt “how it works”, wat bedoel je dan precies?	It’s like how to put these fundamental things next to each other and use them all at once in time, like arrays and functions, fore and attributes and loops and even while and everything. And how to build. We started to learn them one by one. Partly. And now we put them all together…how it works together.	Heeft dankzij GO mentaal model opgebouwd.

Vragen Denken in stappen

vraag	**antwoord**	*nv*	*kenmerken*
Beschrijf hoe je project 3 vanaf het begin aanpakte.	Ah…Mostly it is from Googling. First of all, it is what I do, I search for it you know. I need the topic of my project, I just need to search it. About databases, when you need to find a database…to find a useful database or dataset. And then, for the rest, yeah, I just Google. If I can’t find something, because projects here are like out of nowhere, you don’t know anything at the very beginning. They don’t give you any clues, don’t give you anything from start. You don’t know anything.		Geeft een rudimentaire beschrijving zonder in stappen of deelproblemen te denken.
Dus je begon met het vinden van een geschikte database?	That was the only thing we knew yeah…that we needed a sort of database to start with, to work with. Yeah.		
Hoe ging je verder?	That was really most difficult project ever. We did not get much help from our group and eh… they didn’t say anything about it and we started in the wrong direction. We didn’t know and no one told us about it. We started in a very wrong direction. And we just continued going that way and there as no one to help and last weeks we had switched another way and eh.. it was really a catastrophe, I can say for project 3.		Probeert een trial & error aanpak door veel uit te proberen.
Dus je realiseerde je dat je op de verkeerde weg was. Wat deed je toen?	Yeah eh…we just started with right now, we switched our code and the previous, we just forgot them. We started in a new way and we tried a lot. We tried a lot of possibilities actually. And the right way..was…I mean by right way the other students had chosen. And…they didn’t tell us at the beginning. They told us in general that you can use this and that. And even the wrong way was included in the options that they gave us. But later when we had problems and asked for help, no one was there to help us. But first we were allowed to use that way so yeah, ten days before we delivered the project we started another way.	**L**	Beschrijft hoe het project stagneert. Aanpak wijzigt daarna niet significant.
Wat deed je, toen je begreep dat je niet veel tijd meer had?	I did all my best! I just started from the basic, I asked my friends for help, because they had done most of the project and they explained to me how to start with it.		
Kan je dat specifiek maken?	The part that went most wrong actually, was about the database. How to make use of the database and make the connection in SQL. That was important. So we needed to use a server of the school. And in the beginning I had a sort of local server. Which didn’t work. That’s why I asked from school and making a		Zet kennis van GO in om code te schrijven en ontdekt de

	new connection with the server of the school. Then, just proceeding with the codes. Which were based on the disconnection.		toepassing van het geleerde.
En toen?	Yeah, still we had a lot to code, you know. Because I hadn't done any much search about those codes you know. So I had to just do it and…..		
Hoe ging je op zoek naar informatie over de code?	Ah, yeah it is just like Googling and also asking, here, from the second year students. One who knows and helps. And further, there was a good point about it, because I had to just stay awake 'til five o'clock in the morning to solve one part. I used all the knowledge I knew from Grande Omega. And that was really a changing point of my life I guess, because it worked, finally. I had an error and yeah, I wanted to solve it somehow. And I just did it by myself without Googling, without copying. I did from myself, from my knowledge of Grande Omega.		
Kan je een voorbeeld geven? Hoe hielp Grande Omega jou?	Yeah, with those basics that I knew. I just wrote some codes on my own that I knew. And it worked. The error was solved.		
Dus de code werkte. Hoe ging je verder met je team?	I told in the beginning we didn't really have such a team. It was me and my friend. And she also had no connection with the server, she couldn't do anything, so it was only me. And no one else. Only me…That was a lot to do…And I can't explain more but we just went further you know…With big help from people and just checking the code of other samples and try to simulate them in my own code.		
Hoe hield je het overzicht over wat je moest doen?	At first we couldn't do it because we lost the track. Only I can talk about the last ten days and that was like...honestly I don't know anymore, I don't remember, there was too much pressure (laughs). It was just ten days, it wasn't much for doing it and I barely remember what happened. It was just solved, that's it. I don't know how (laughs).		Geeft geen specifieke voorbeelden
Hoe zorgde je ervoor dat je niets vergat?	It wasn't that difficult. First of all we have a sort of, they say modulewijzer. Every time we miss something we just go and check there, And oh, I just finished one part and one day I came to school and then I showed it to my friends and they told me, now you are missing something. For delivering the project and to get your grade, you need this also. Seriously? Yeah. Then I know it wasn't finished. I missed something and I had to do that part also. That is one of the examples that happened.	**L**	
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en waarom je dat zo deed.	No, we didn't have Scrum honestly, because we were only two. And we just checked the program. And I had the program so no one could check it either. Only my friend, when we were at school, we could just work on it together. We did a lot of.. we checked it every time you know. Every single code that you write you have to check if it is working good like that. And once we just faced an error, sort of bug, and then we knew we just had to solve it. And yeah, we solved it. That error, I told you, it took me until five in the morning.	**L**	Beschrijft geen revisies of iteraties vanuit Scrum.

Gebeurde dit nogmaals; een revisie, het opnieuw proberen?	No, no.		
Was er een moment dat je het met de product owner moest bespreken?	Yeah, actually, we were, how do you say it, “lopen achter”, we didn’t have the same tempo of the product owner. She knew it also that we were not at that stage enough. And she just warned us that we should be ready for the time of delivery: You haven’t done much so far. We just told her about the problems with SQL and the server like that. Here’s the thing: the product owners are not really qualified, they don’t have enough information about programming. So they really can’t be of any help.		
Op welke manier heb je een complex probleem zo geformuleerd dat het met code op te lossen was?	What do you mean problem…We had to solve it with code. There was no other way, no. It is all bout code and programming.	**L**	Geeft geen voorbeeld.
Welke feedback kreeg je?	Actually, we got 7.3 . It was deserved better than this. We needed to have two variables. The visualisation was small, the font was too small. It wasn’t a big deal you know.		
Op welke wijze helpt het oefenen in Grande Omega bij de oplossingsrichting?	There was a sort of bug that we had to solve. I just Googled at first by the way to see how I can solve this and I tried a lot of things but it didn’t work via Google. I had the idea what was really wrong and to find out where does it come from, this error. We did a lot of Grande Omega practicum. So many things are just like fixed in our mind. It is like, yeah, something you learn in your childhood because of repetition and then you just remember it forever. Something like this was happening and you know, sometimes so many samples of Grande Omega come in front of your eyes. You see it and then I just tried one of them you know and I remembered we had such a thing before in coding in Grande Omega and I tried the basics, what I knew about the basics, about for loop. I used it. I tried to write code in that way. Using for loop and basics and the model that I had in my mind, the pattern I had in my mind from somewhere in Grande Omega. And it worked.	Beschrijft hoe door tijdsdruk en spanning een mentaal model dankzij GO werd geactiveerd, waardoor de oplossing binnen handbereik lag.	

Vragen Abstraheren

vraag	**antwoord**	*nv*	*kenmerken*
Wat waren de belangrijkste onderdelen van project 3?	O yeah, actually we needed to have two forms, two graphs, and there should actually be a sort of filter. That was very important. To have some sort of filters that you could filter between the options variables and just choose and let them to be shown on the graph of the diagram.	**M**	Isoleert de belangrijkste informatie.
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?	The number of filters, how many filters you need to have. I could just decide how big my database should be. I had a very big one at the very beginning but I started to delete so many columns and rows, to make it easier also because they weren’t important. It is not the good way to do it, it was just to make it easier. The good way is to leave it the way it is and all the	**M**	Geeft een voorbeeld om relevante en minder relevante kwesties te scheiden.

	variables. And put all the columns and rows in your….That would be very nice but I wanted to make it easier, just something deliverable.		
Op welke wijze helpt het oefenen in Grande Omega hierbij?	To be honest, no. It has nothing to do with Grande Omega. No, not at all.	Ziet geen relatie tussen GO en het project.	

Vragen Algoritmisch denken

vraag	**antwoord**	*nv*	*kenmerken*
Hoe zet jij een probleem om in code?	You want me to explain how I code? I will make a class. That makes everything easier. First thing for sure is making a class. The problem I believe should be solved inside the function. You just define some attributes of fields for that class, what you need to have, as instruments and with functions. I think about the functions, what I can do inside, just to think about how to solve that problem inside the function. That is the easiest and very basics of solving problems in code. The next step depends on the sort of problem it is. It is all about the functions and finally at the end I put a variable and use that class and function inside that variable and try to print it, you know. Somehow just test if it is in a good way, if that is something that I want. We are still learning how to test… It is not really a good way that we are learning. I can honestly say that I haven't learned anything from testing. But I know a lot. It's about conditions, false and true. Sometimes you have in one condition true and false or conditional tables to see how many possibilities you have, how many should be true and how many should be false. That is one of the ways. Honestly, practically, I haven't used them. Not yet.	**H**	Beschrijft specifieke stapsgewijze aanpak en zet denkstappen uiteen. Heeft kennis van testen, benoemt kenmerken en beschrijft aanpak, maar past kennis nog niet toe.
Hoe probeer je uit of je code werkt?	Yes I told you, I just use some different variables, define variables and then I put them inside an object and try to print it. That is the way I do. We have a sort of test like that. Some border testing you know. Variables which are at the border. Numbers which are at the border or just random numbers. These are a sort of testing.		
Is dit de wijze waarop je checkt of je code correct is?	If there is really something wrong with the code you will see it through visual studio. Through the program software. That is a very obvious one, you will see you did it wrong. But if it runs, you see some results as a form or app and you see you got something. But then you have to look for the bugs. There were tiny errors that you have in the middle of the program. And then you need to check for that. It depends what sort of problem you have and what sort of project. Like an app, you really need to give it to people to test.	**M**	Beschrijft een algemene aanpak.
Hoe vind je die kleine fouten?	If I know there is something really wrong, if I am aware of it, I will use the debugging thing. Just to run it and find the step that I am suspicious, I will check what variables it has, to see if it's going good.		Controleert code door te testen met behulp van debugging tools.
Wanneer wantrouw je de code? Kan je een voorbeeld	Sometimes it is difficult. I am not really a programmer, you know. But I have no big programs		

geven?	you know, so I really can't talk about this case. But last time I wrote a very simple program and I knew it was wrong. I got an answer but regarding the problem I had, I am not getting what I have to. I'm doing something wrong. You have to solve the way to get to that answer. And sometimes you can't do it.		
Hoe ga je na of jouw code correct functioneert? (Als je merkt dat jouw code niet goed werkt, hoe ga je dan te werk?)	Mostly I use tools, Internet, But it doesn't solve all the problems. Then I ask the second year students, because they are good at programming. Yes, peer coaches, but not all of them. Just one of them. The rest don't know. They are not available and they have not the knowledge.		
Vraag je ook hulp aan de docenten?	Generally speaking about coding, they don't help, to be honest. Or they didn't have time or…they don't have much knowledge about C#. And they come and have a look and just say: I don't know! You solve it. They don't know what is going wrong and that's it. You can't count on them (laughs).		
Hoe zorg je ervoor dat jouw code begrijpelijk is voor anderen?	Not that much. But in the the project I gave my part to my friends. They needed this one to use…I had solved some problems and they could use it to solve their own problems, as a pattern. I don't make sure that they can work with my code. My friend understood. You know, we are at the same level so my code is very easy to understand for sure. It's not very complicated. It is just like reading a very simple book (laughs).	**L**	Beschrijft geen kenmerken. Noemt geen aandachts-punten.
Heb je wel eens teruggekeken naar je oude code? Begreep je die toen nog?	No, because we didn't make any codes in the beginning of this year. I did just two very small codes very recently. The very first programming I did, was for Analysis. The first in my life.		
Op welke wijze helpt het oefenen in Grande Omega hierbij?	A lot. I didn't expect it myself. I thought Grande Omega is not going to do anything. But when I first started programming, some views, visions of the code came to my mind, Because it was repeated a lot of times, I could just write it out of my head, without thinking…maybe I forgot just one something. The whole idea was there in my mind.	Beschrijft hoe (onbewust?) opgebouwd mentaal model beschikbaar bleek.	

Transcript interview 2 _onderzoek Grande Omega OP4 2018

Tijdstip: dinsdag 5 juni 2018

Plaats: HR Rotterdam

Interviewer: AvB

Geïnterviewde: 2, studentcode 69

Algemene vragen

vraag	**antwoord**	*kenmerken*
Wat is je vooropleiding?	Ik heb hiervoor mbo-4 ICT-beheerder gedaan.	Mbo.
Omschrijf je programmeer-ervaring. Wat kon je al toen je begon aan dit collegejaar?	Ik kon al webprogrammeren, websites bouwen… en de achtergrond met PHP.	Voorkennis in programmeren.

Wat deed je toen allemaal?	Ik had op mijn stage een app gebouwd. Dat noemen ze een configuration management database. Die hield bij wie welke laptop heeft, wie welke telefoon heeft en alle bruikleenovereenkomsten, die werden digitaal bewaard. Die had ik naar de gebruiker verstuurd. De opleiding ging niet over programmeren. Dat had ik mezelf geleerd op stage en in m'n vrije tijd. Ik had op stage een database gemaakt en uiteindelijk hadden ze een heel systeem.	Deels autodidact.
Als je jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?	Ik zou mezelf, denk ik, een 4 geven. Ik kon toen al programmeren maar het was, zoals ze dat in de ICT zeggen, spaghetticode. Enorm lange code, dan moest je heel veel kopiëren en plakken, dat zag er niet echt heel netjes uit. Het werkte wel uiteindelijk, na heel veel testen…Veel bug fixes.	Maakte eerst spaghetticode. Van 4 naar 7.
Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden na dit collegejaar, welk cijfer is dat dan? Om welke reden?	Een 7. Omdat ik het afgelopen jaar heel veel onderzoek zelf heb gedaan naast de vakken van eh…hoe doe ik dit, hoe doe ik dat. Voor m'n portfolio heb ik zelf een hele website gebouwd, die er veel beter uitziet dan die website van m'n stage.	Geleerd hoe code op de achtergrond werkt. Kan geleerde toepassen.
Wat ziet er dan beter uit?	Door onderzoek te doen naar van: hoe kan ik dit het beste aanpakken. En de kennis die ik heb opgedaan bij Development. Voor bepaalde dingen, waarvan ik dan denk, oh zo werkt het, dat kan ik dan zo toepassen. Zo heb ik voor een deel zelf onderzoek gedaan en van Development geleerd. Hoe de code op de achtergrond werkt, ah…zo werkt het. Als je dat weet dan kan je het zo toepassen	50% GO, 50% zelfstandig onderzoek (ook via DEV) en projecten.
Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?	Voor mij wel aardig wat. Ik denk voor programmeerkennis ongeveer de helft, 50%.	
En die andere 50%?	Heel veel zelf onderzoek gedaan. Van projecten heb ik ook heel veel geleerd.	

Vragen Denken in stappen

vraag	**antwoord**	*nv*	*kenmerken*
Beschrijf hoe je project 3 vanaf het begin aanpakte.	In het begin ging het niet zo heel soepel. We begonnen met oké, wat willen we gaan visualiseren, wat voor data hebben we daarvoor nodig. En ik wilde het aantal studenten met leeftijd visualiseren. Toen zijn we daar data voor gaan zoeken. Toen zijn naar het CBS gegaan en hebben we daar een dataset gevonden. We zijn helemaal gaan uitzoeken hoe we die data wilden hebben. Je kon aangeven wat wel, wat niet. Toen hebben we de dataset gedownload, meerdere datasets, en heb ik een database aangevraagd bij de server en geïnstalleerd en alle datasets daarop gezet. Dat kende ik al van mijn vooropleiding. Bij ICT beheer moet je de hele dag databases beheren.	**H**	Beschrijft een stapsgewijze en volgordelijke aanpak.
Hoe hield je het overzicht	We gebruikten Scrum. We hadden een Trello board	**H**	Beschrijft een

van wat je moest doen bij dit grote project?	waarop stond van oké hier zijn we nu bezig, dit moeten we nog doen, dit hebben we al gedaan. Dat verspringt dan. We hadden ook een Scrum document met een product backlog waarin alles staat wat we voor dit product moeten maken. Per backlog, per sprint, een user story, taken, prioriteiten waar het aan moet voldoen. Zo hadden we een beetje overzicht. Die Scrum meetings, dat hadden we 2,3 per keer per week. Sowieso elke woensdag , dan hielden we maandag nog eentje en soms in het weekend.		gestructureerde aanpak met behulp van Scrum. Noemt deelproducten.
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en waarom je dat zo deed.	Reviews? Oké, waren er nog slechte punten, goeie punten om op te schrijven. Daar deden we in het begin niet zo heel veel mee. Aan het einde van het project hadden we een soort van kaart gemaakt van slechte punten, de verbeterpunten daarvoor en nog ergens de goeie punten. En hoe we dat door moeten gaan zetten.	**M**	Rubriceert acties van proces.
Op welke manier heb je een complex probleem zo geformuleerd dat het met code op te lossen was?	Je gaat eerst kijken, hoe moet het er ongeveer uit gaan zien, dus een storyboard bouwen. Willen we een grafiek met die data erin, en dan ga je opzoeken: hoe krijg ik dingen in een grafiek en dan ga je zo stap voor stap proberen alles op te lossen. Kleine stapjes. Eerst met nepdata werken en uiteindelijk lukt het je om de data uit de database te krijgen en in een grafiek te stoppen. Dan kan je er een applicatie omheen gaan bouwen. Dan ga je op zoek hoe filteropties werken, hoe je dat kan bouwen.	**H**	Beschrijft een stapsgewijze en volgordelijke aanpak. Maakt gebruik van schematisch hulpmiddelen.
Stap voor stap oplossen zeg je, welke stappen kan je noemen?	Eerst kijken, oké, ik heb een grafiek. Dan ga je op Google opzoeken, oké, hoe krijg ik data in die grafiek via C#. Oh, zo ongeveer…en dan komt het in een grafiek te staan en dan ga je zo verder. Hoe krijg je dus data uit een database, daar is een functie voor. Hee, ik krijg er data uit! En dan ga je dat aan elkaar combineren.		
Op welke wijze helpt het oefenen in Grande Omega bij de oplossingsrichting?	De kennis was daar vooral, hoe classes werken…dus ook weer de achtergrond van de code.	Vindt dat GO helpt bij het maken van classes.	
Gebruikte je dat ook bij dit project?	Ja, want we moesten in C# werken en dat is een programmeertaal waar je in classes moet gaan werken, anders doet ie het niet.		

Vragen Abstraheren

vraag	**antwoord**	*nv*	*kenmerken*
Hoe pakte je project 3 aan? Hoe organiseerde je bij project 3 de informatie die je nodig had om te kunnen programmeren?	Experimenteren, werkt deze grafiek op onze data? Beetje logisch nadenken van hm.. we krijgen heel veel kolommen, als we nu een verticale grafiek doen, die verticale lijntjes heeft, horizontaal sorry, en dan ga je verder kijken, en dan kom je op de goeie uit.	**M**	Beschrijft een experimentele werkwijze met behulp van logisch redeneren.
Maak je daarbij ook gebruik van andere bronnen?	Vooral gewoon zelf testen.		
Hoe deed je dat?	Dan veranderde je de grafiek zo van, oké, ik heb nu een staafdiagram, naar een cirkeldiagram en dan ga je zo verschillende diagrammen af. Kijken van ziet het er goed uit, is het logisch, je kijkt of het nog leesbaar is. Als je zo´n (gebaart iets kleins) grafiekje hebt en je hebt 300 kleine lijntjes, dan is het niet echt		Isoleert de essentiële

	leesbaar!		informatie maar beschrijft niet hoe.
Wat bedoel je, dat het logisch is?	Stel je voor, je hebt een staafdiagram en een staafje is 300 eenheden hoog en in een cirkeldiagram doe je hetzelfde maar dan met 100 en maakt hij er een raar teken van. Zoiets. Dat zulke dingen niet gebeuren.		
Als je naar het project zelf kijkt? Wat zou je als belangrijkste onderdelen van het project noemen?	De belangrijkste onderdelen….Project drie…Toch wel dat je moet gaan visualiseren en de juiste datasets zoeken, anders krijg je geen goede grafieken. Die data die je eruit krijgt, moet je in een grafiek stoppen.	**M**	Maakt gebruik van toetscriteria als checklist voor een voldoende en om de essentie te isoleren.
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?	Belangrijk is sowieso waar je de punten voor krijgt. De toetscriteria. En dan ga je kijken, goed, oké, wat moeten we doen om een voldoende te krijgen, wat halen om er punten voor te krijgen. We willen dit, dit en dit en dan moet dit voldoende worden.		
Waren de toetscriteria dan een hulpmiddel om ervoor te zorgen dat je wist wat belangrijk was?	Dat was belangrijk. Was design belangrijk of was het belangrijk om data gewoon visueel te hebben en een hele simpele applicatie te bouwen? Ik wist dat design niet bij de toetscriteria stond. Het was minder belangrijk om een heel mooie applicatie te bouwen.		
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Bij die onderdelen heeft Grande Omega eigenlijk niet echt geholpen. Ja, niet.	Ziet geen relatie tussen GO en project.	

Vragen Algoritmisch denken

vraag	**antwoord**	*nv*	*kenmerken*
Hoe zet jij een probleem om in code?	Da's een goeie. Ligt helemaal aan het probleem. Vaak ga ik eerst kijken, oké, ik heb een probleem….even kijken, een voorbeeld. Voor portfolio wilde ik een aanvraagsysteem maken dat mensen een account kunnen aanvragen. Dan ga ik kijken, wat moeten ze daarvoor doen. Moeten ze daarvoor iets invullen, dus tekstboxen moeten dan gemaakt worden - weet ik al- informatie in de tekstboxen moet worden verstuurd naar mijn database, dan moet dat opgeslagen, die aanvraag. Vervolgens moet ik een e-mail krijgen, van hee, er is een aanvraag binnen en dan moet ik dus een pagina met die aanvraag gaan controleren. Wie is het, wat wil ie, en waarom. Vervolgens moet ik de aanvraag kunnen goedkeuren of kunnen afkeuren. Bij goedkeuren moet er een account aangemaakt worden voor diegene. Hier is je account en zijn je gegevens, hier kan je je wachtwoord resetten. Dat zijn alle stappen en moet ik dus stap voor stap gaan kijken hoe ik dat moet doen. En dan in code om gaan zetten. Of opdelen van problemen. Als je een groot probleem hebt, krijg je het nooit opgelost.	**H**	Beschrijft een stapsgewijze aanpak in logische volgorde. Zet een groot probleem om in deelproblemen (DIS)
Hoe ga je na of jouw code correct functioneert? Hoe ga je dan te werk?	Vaak ga ik 'm dan testen. Zo van oké, als ik deze fout invoer, geeft hij dan ook een fout terug, ja of nee? Als ik het goed invoer, werkt ie dan goed? Als ik het een paar keer doe, werkt ie ook nog goed? Werkt ie op mijn laptop dan ook nog goed?	**H**	Maakt gebruik van testen op meerdere platforms.
Hoe doe je dat, dat testen? Wat is je manier of wat zijn je tools?	Ik doe vaak wat ik bij het mbo geleerd heb. Ik heb geleerd als ik iets moet testen om op drie waarden te testen, dus onder je waarde, dat je een error krijgt, op		Test op verschillende waarden, benoemt

	de juiste waarde die net goed moet gaan, en daarboven. Geen idee hoe dat heet.		testmethoden.
Nog meer testmethoden?	Die zijn we aan het leren, toevallig deze periode. Decisions table, met alle opties van juist en onjuist. Grenswaardenanalyses, net als daarnet, lijkt een beetje op de grenzen zitten. Dus alles eronder, erop en erboven. Equivalentieklassen, dus dan pak je waardes tussen waardes in, en dan ga je daarin een paar waardes testen. Tijdens de uitkomst ga je kijken wat je verwacht.		
Heb je wel eens teruggekeken naar je oude code? Wat zou je nu anders doen?	Meer functies bouwen. Op stage heb ik ergens een SQL-verbinding gedefinieerd en continu overal gecopyd-pasted naar elke pagina toe, dus als er ooit een keertje iets in verandert, dan moeten ze elke pagina het wachtwoord gaan veranderden als hij wordt gereset (lacht). Dat soort dingen Nu heb ik gewoon een functie, die kan ik in een keer aanpassen van hey, klaar.	**M**	Herkent fouten en beschrijft een verbeterpunt.
Hoe zorg je ervoor dat jouw code begrijpelijk, werkbaar is voor anderen?	Door overal zo'n beetje comments neer te zetten. En als het eenmaal werkt, dus ook wat netter te maken dus dingen in functies zetten, dus als er niet allemaal dingen van 300 tekens staan, het ook gewoon een functie kan zijn die je aanroept. Het zijn vaak heel lange codes en het wordt echt spaghetticode als we ook nog code to gaan gebruiken. Dan zet je code to en dan ga je terug naar het begin van je code ergens en dan vliegt je code alle kanten op, continu. En dan is het voor niemand meer te volgen.	**H**	Noemt twee kwaliteits-kenmerken. Beargumen-teert de noodzaak van kwaliteit.
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Om de code te begrijpen, wat het doet op de achtergrond. En als iets fout gaat, dan kan je echt stap voor stap door de code heenlopen uit je hoofd en dan weet je oh, daar gaat ie fout. Zo helpt het mee. Ik had vroeger nog wel eens dat ik hele dagen met een probleem zat, en nu veel minder gelukkig,	Leert van GO wat de code op de achtergrond doet.	
Hoe doe je dat, stap voor stap door de code lopen?	Dan ga je echt per regel van oké, wat gebeurt hier hoe ziet me geheugen er dan uit en dan de volgende stap debuggen als dat lukt bijvoorbeeld, dan zie je ook het geheugen zelf en elke stap die de code dan maakt, voer je zelf dan ook uit.	Loopt zelf de code stap voor stap, regel voor regel na.	
En zo gebeurt het ook in Grande Omega?	Daar ga je ook stap voor stap doorheen ja.	Herkent een stap-voor-stap opbouw in GO.	
Wil je zelf nog iets kwijt, iets vertellen over deze onderwerpen, die we net hebben besproken?	De onderwerpen zelf niet echt. Voor Grande Omega, daar heb ik wel verbeterpunten voor. Het gebeurt heel vaak dat we backward assignments aan het maken zijn, dan moet je dus gaten in de code invullen. Dan krijg je heel vaak een vage error code, dus iets als expected IOF-file system nog wat en dan column, regel 46, column 47, die moet ik allemaal gaan tellen en dan kom ik erachter dat er maar 40 regels zijn. Ik heb geen flauw idee waar het fout gaat. Dat zijn een beetje van die vage error-codes waar je zelf...je krijgt feedback maar je hebt geen idee wat het is. Er zijn van die dingen waar je vast in blijft lopen en dan moet je naar practicum, want je komt er gewoon niet uit. Er was laatst een docent die zei: Ik kijk wel even in de achterkant, want ik zie het zelf ook niet.	Heeft behoefte aan inhoudelijke feedback.	

Transcript interview 3 _onderzoek Grande Omega OP4 2018

Tijdstip: dinsdag 5 juni 2018

Plaats: HR Rotterdam

Interviewer: AvB

Geïnterviewde: 3, studentcode 52

Algemene vragen

vraag	**antwoord**	*kenmerk*
Wat is je vooropleiding?	Ik heb hiervoor mbo scheeps- en jachtbouw gedaan. Daarvoor heb ik een vmbo-opleiding metaalbewerking gedaan.	Vooropleiding mbo.
Omschrijf je programmeer-ervaring. Wat kon je al toen je begon aan dit collegejaar?	Ik kon aardig programmeren al. Ik denk dat ik al 6 á 7 jaar ervaring had met programmeren. Maar dat was wel voornamelijk alleen het praktische deel, dus niet de theorie daarachter. Dus waarom sommige dingen moesten.	Voorkennis in programmeren: 6 á 7 jaar ervaring als autodidact in praktische toepassing. Geen theoretische kennis.
Wat deed je toen allemaal? Voor werk of plezier?	Eh…voornamelijk websystemen en zulk soort dingen. Nee, puur hobbymatig.	
Als je jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?	Ik zou het een 5 geven. Ik kon op zich bijna alles wel maken wat ik in gedachten had, maar het was nooit de efficiëntste of de beste manier om het te maken. Dat zat voornamelijk in de kennis die ik had. Zoals ik zei, ik kon het, maar ik wist de theorie erachter niet dus. Vaak was er een veel betere oplossing als je wist hoe het werkte. Dus vaak gebruikte ik een omweg voor iets wat al bestond.	
Hoe heb je dat jezelf dan geleerd?	Voornamelijk filmpjes kijken, op internet zoeken en blijven, blijven doen.	Van 5 naar 5,5.
Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden na dit collegejaar, welk cijfer is dat dan? Om welke reden?	Dan zou het een 5,5 zijn. Ja, een halfje erbij. Dat komt voornamelijk door de projecten die ik met klasgenoten heb gedaan en de kennis die ik van hen erbij heb gekregen, zeg maar.	
Wat voor kennis heb je erbij gekregen?	Nou, de manier om dingen mooi op te splitsen zodat je veel overzichtelijker een groter project kan maken. Dat voornamelijk.	Heeft geleerd om code overzichtelijk op te splitsen.
Wat splits je dan op?	Als je code schrijft, dan kan je een hele lap tekst maken en telkens iets herhalen. Bijvoorbeeld voor gebruikers op te halen, ik noem maar wat. Maar je kan ook een mooie component daarvoor maken en vervolgens dat component aanroepen. Dus dan splits je de code op om het voor jezelf overzichtelijk te maken en makkelijker uit te breiden.	
Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?	Als ik heel eerlijk ben 0%. Hoe Grand Omega momenteel in mekaar zit, vind ik niet dat je er echt wat van leert. Je bent meer bezig met hoe Grande Omega werkt en wat zij van je verwachten in plaats van dat je bezig bent met programmeren. Een groot deel van het programmeren is ook dat je creatief kan	0% GO Alles geleerd van klasgenoten, peer coaches en Internet.

	zijn in hoe je iets maakt en Grande Omega staat dat momenteel niet toe. En dat is heel nadelig.	
Aan wat had je het wel te danken?	Voornamelijk aan de kennis die klasgenoten al hadden en het samen uitzoeken en samen te gaan werken. En natuurlijk op internet hoe je dingen op moet lossen. En desnoods hulp vragen aan de peer coach, mocht je er echt niet uitkomen. Eh, het is dat ik beneden mijn niveau begon en dat het dit nog steeds is, maar het vak Development zelf, op zich kan je daar ook aardig wat informatie uithalen. Ja, voornamelijk de basis en toch de theorie daarachter, die heel belangrijk is.	
Wat bedoel je met basis?	Hm…ja de manier waarop je code schrijft bedoel ik dan. Je moet gewoon ..nou ja moet…je kan het op een bepaalde manier schrijven en er zijn heel verschillende manieren om dat te doen ja, maar het moet wel netjes zijn.	Wil leren door opdrachten te maken en daar feedback op te krijgen.
En wat bedoel je met theorie?	Ja, de manier waarop dingen op worden geslagen en hoe een array in een lager niveau werkt dan in de code die je schrijft. Dat is wel handig om te weten als je het schrijft. Waarom iets gebeurt, laat ik het zo zeggen.	
Welke andere dingen leer je bij DEV, anders dan in Grande Omega?	Het grootste nadeel van Grande Omega leren is…als je wilt leren door dingen te doen. Ik zou het liefst gewoon leren door opdrachten te maken en daar feedback op te krijgen. Momenteel, als je iets invult dan is het gewoon - bam - fout. Er staat dan fout of goed en je kan niet leren om opdrachten te maken als er alleen maar staat fout of goed. Dan weet je wel dat het fout is, maar je weet niet wat er fout is. Het voordeel van DEV is, dat je gewoon een vraag kan stellen aan een docent en die legt het uit en die gaat net zolang door - of neemt je later apart - tot je het snapt.	Bij DEV-practicum geleerd om code te begrijpen.

Themavragen Denken in stappen

vraag	**antwoord**	*nv*	*kenmerk*
Beschrijf hoe je project 3 vanaf het begin aanpakte.	Ik kwam in het project (in week 2) en de meeste beslissingen waren al gemaakt dus de taal waarin het gingen doen, wat we daarbij gingen gebruiken. Dus voornamelijk was het voor mij meer van: oké we gaan het zo doen, dit moet gebeuren, dus we gaan het zo doen. Een deel van de dingen was, dat we informatie moesten hebben om in het programma te gebruiken, de data, zeg maar. Die hebben we opgehaald en toen kwamen we erachter dat een deel alleen live te zien was en daar heb je niet zoveel aan als je het wil gaan vergelijken met elkaar dus hebben we heel de periode, zo ongeveer en maand lang, hebben we al die informatie opgeslagen om het vervolgens in het eindproduct te verwerken.	**L**	Geeft rudimentaire beschrijving maar geen details over de aanpak.
En toen?	We hebben het met een andere dataset vergeleken. De bezetting van parkeergarages, dat hebben we opgeslagen, elk uur van de dag, voor een maand lang. Toen hebben we het uiteindelijk vergeleken met het weer van dezelfde dagen op dezelfde tijd.		

Hoe hield je het overzicht van wat je moest doen bij dit grote project?	U zegt groot project. Ik vond het een vrij klein project, moet ik zeggen. Als ik heel eerlijk ben: 10 uurtjes in me eentje eraan werken en toen was het ook klaar. Maar dat komt dan meer door de voorkennis en omdat je met mensen bezig bent die geen voorkennis hebben, dus die moet je helpen. Maar eh…ja, voornamelijk gewoon door de Scrum projectmethodologie die we gebruiken. En daar werd gewoon ingezet wat er nog moest gebeuren om tot het product te komen dat we wilden hebben. Voordat we begonnen aan die week besloten met z'n allen wat er die week moest gebeuren. We gingen een uurtje, kwartiertje - ligt eraan hoelang we nodig hadden - even met heel de groep om tafel zitten en dan van joh, dit moet gebeuren of dit kunnen we doen in deze week. Dat hielden we bij in Trello, dat is een speciaal Scrumboard online.	**M**	Schetst een algemeen beeld van Srcum-methode
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en waarom je dat zo deed.	Ja, er waren reviews, meer om te testen of alles goed ging. Eigenlijk na iedere sprint, om de twee weken als je ergens aan hebt gewerkt, na iedere sprint gingen we kijken van wat kunnen we doen in de volgende sprint en wat gaan doen voor de volgende sprint. Een nieuwe planning maken, dingen die aangepast moeten worden of niet goed zijn of overnieuw moeten.	**M**	Maakt geen onderscheid in toetsen project en toetsen code
Op welke manier heb je een complex probleem zo geformuleerd dat het met code op te lossen was?	Er moest wel iets gemaakt worden en je begint met niks. Maar op zich, informatie ergens vandaan halen en met mekaar vergelijken is niet heel veel werk. Althans, je kan alles automatisch laten doen dus je schrijft een soort algoritmetje en dat zorgt ervoor dat alles uitgerekend wordt met de informatie die je invult. Dus eigenlijk ja, zodra je de informatie ergens in heb gezet, hoef je het alleen maar uit te lezen, een stuk code te schrijven om met mekaar te vergelijken. Dat is het.	**L**	Schetst een aanpak zonder specifieke stappen.
Hebben jullie zelf code geschreven?	Ja, ja, we hebben alles zelf geschreven. Wij waren met vijf man bezig met het project. Daarvan twee zonder programmeerervaring en eentje die aardige programmeerervaring had. De andere twee deden vooral programmeerwerk, daar was ik er een van.		
Op welke wijze helpt het oefenen in Grande Omega bij de oplossingsrichting?	Nou, op dat moment echt heel weinig want de taal die school ervoor had geschreven, was C# om in te werken en Grande Omega was op dat moment, als ik het me goed herinner, nog steeds bezig in een soort van eigen verzonnen taal die goed op alles aan zou sluiten maar geen C#. Dus je bent in een taal bezig waar je van school uit nog nooit iets van geleerd hebt. Althans van Grande Omega uit, nog nooit iets geleerd hebt, laat ik het zo zeggen. Dus op zo'n moment heb je er vrij weinig aan. Ja, je hebt een deel van de theorie van ja, ik weet hoe ik statements moet maken en dingen moet vergelijken. Maar je weet nog steeds niet hoe je het opschrijft.	Vindt dat taal van GO niet aansluit op C#. Heeft deel van theorie begrepen maar kan kennis niet toepassen.	

Themavragen Abstraheren

vraag	**antwoord**	*nv*	*kenmerk*
Hoe organiseerde je bij project 3 de informatie die je nodig had om te kunnen programmeren?	Een soort van korte samenvatting dus. Ja eh.. informatie ergens vandaan halen, opslaan en vergelijken. Of uitlezen en vergelijken.		Kan de essentiële informatie niet isoleren.
Als je naar het project zelf kijkt? Wat zou je als belangrijkste onderdelen van het project noemen?	Pfff…de belangrijkste onderdelen, op de manier van hoe we het uiteindelijk gedaan hebben? Oké. Voornamelijk de voorkennis die mensen al hadden in de projectgroep en… Internet.	**L**	
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?	Zodra er een functie gemaakt moest worden ging ik wel kijken, …als we het nu niet zouden doen, werkt het programma dan niet meer. Of haalt het z'n doel niet meer, want zodra dat het was kan je het beter achteraan zetten. Is die op dit moment nodig? Als we 'm laten zitten werkt alles dan nog en zo ja dan sloegen we hem over en deden we hem later. We hadden een keer in de zoveel tijd gesprekken met de owner, oftewel de klant. En de klant geeft ook wel aan wat graag de volgende stap zou zijn, voor de klant zelf.		Geeft geen voorbeelden waaruit het onderscheid tussen relevant en minder relevant blijkt.
Volgden jullie die volgende stap?	Eh…ligt eraan. Als de volgende stap niet essentieel zou zijn of het voegt niet heel veel meerwaarde toe, en tegelijkertijd moest er nog iets gebeuren wat wel heel belangrijk was voor het project, en dan deden we eerst het belangrijke natuurlijk. Dan probeerden we dat ook uit te leggen aan de klant van joh, kunnen misschien beter eerst dit doen want dit is gewoon veel belangrijker. We waren bezig met – om… in het geval van de parkeergarages - wilden we onderscheid maken tussen de verschillende parkeergarages en daar hadden we in het begin vrij lelijke knoppen voor en dan wou de klant graag dat die knoppen mooi gemaakt werden, maar dat was nog niet heel belangrijk, aangezien je de tijd nog niet kon selecteren. Dus dan heeft de tijd op dat moment voorrang want dat is voor de functionaliteit van het programma, dus niet alleen iets wat er mooi uitziet.	**L**	Probeert wel onderscheid te maken tussen belangrijk en minder belangrijk maar gebruikt geen systematiek of schematische hulpmiddelen.
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Oh, nul. Niet, nee.	Ziet geen relatie tussen GO en het isoleren van essentiële informatie.	

Themavragen Algoritmisch denken

vraag	**antwoord**	*nv*	*kenmerk*
Hoe zet je een probleem om in code? En hoe ga je na of jouw code logisch is?	Iedere mogelijke optie die er is: testen. Voornamelijk door overal op te klikken. Als je bijvoorbeeld een e-mail in moet vullen door de punt weg te laten of het apenstaartje weg te laten. Gewoon zoveel mogelijk opties, gekke dingen die je maar kan bedenken altijd proberen. En zodra ik het uit zou willen geven wat ik heb gemaakt, dan ga ik gewoon aan kennissen en vrienden vragen van joh, kan je het testen. Om de een of andere reden krijgen ze het altijd voor mekaar om het alsnog te breken!	**M**	Schetst een algemene aanpak. Laat het door derden testen.

Hoe probeer je uit of je code werkt?	Een voorbeeld was voor het web systeem dat ik heb gemaakt. Dat ze op een link drukten en de link was bijvoorbeeld 50 pixels hoog. En daarin stond de tekst en ik klikte telkens op de tekst. En zij klikten net onder de tekst en daarin werkte de link niet meer. Dat was gewoon een hele stomme fout. Dat alleen de tekst de link is maar ja, oh, dan dacht ik, dat werkt toch niet allemaal goed, dat moet ik aanpassen. Dat zijn dingen die je zelf niet snel zou doen want je weet hoe het werkt en ja, voor jou is het logisch om dan op die plek te klikken, maar voor iemand anders niet.	**M**	Noemt een algemeen voorbeeld van trial & error. Noemt geen specifieke aanpak, oplossing en controle.
Hoe ga je na of jouw code correct functioneert? Als je merkt dat jouw code niet goed werkt?	Ligt eraan hoe erg het is. Als de code heel slecht is dan gooi ik alles weg.	**L**	Noemt geen specifieke aanpak, oplossing en controle.
Heb je wel eens teruggekeken naar je oude code? Begreep je die toen nog?	Ik heb het wel eens gehad dat ik een projectje van een jaar geleden bekeek en toen dacht ik: laat ik dit weer eens oppakken. Toen zag ik: dit slaat echt helemaal nergens op. Weg ermee, ik begin overnieuw.		
Hoe zorg je ervoor dat jouw code begrijpelijk is voor anderen?	Dat heb ik van klasgenoten geleerd en door in projecten samen te werken geleerd. Je moet gewoon alles heel goed benoemen, goed nadenken over de functie, benoemen hoe je het opschrijft, of je het goed verdeelt om alles zo netjes mogelijk te maken. Zodat als je er later in terugkijkt, dat je nog steeds snapt wat er gebeurt en niet dat je eerst uren moet nadenken over: wat gebeurt hier. Dan is het nodig om documentatie erbij te schrijven. Een voorbeeld is gewoon een korte beschrijving van de functies en classes die je in je code hebt staan. Zodat ze toch niet eerst heel je code moeten nalezen van wat doet dit. Het is gewoon deze benaming, oh, die doet ongeveer dat. Dan weten ze veel sneller waar ze moeten kijken en is het veel makkelijker voor iemand om het op te pakken.	**H**	Heeft geleerd om te verdelen en zorgvuldig te benoemen. Heeft geleerd om lange code op te splitsen. Werkt met classes en functies.
Nog andere methoden?	Je kan er comments bij schrijven dus eigenlijk een soort van documentatie, maar dan in je code zelf. Ook best handig voor jezelf om te doen, soms.		
Wat doe je nu anders?	Voornamelijk het net schrijven. Netjes verdelen. Dat zijn voor mij wel de belangrijkste veranderingen geweest. Wat netjes verdelen betekent? In het begin gooide ik alle functies in een document, lekker makkelijk, dan weet je waar je moet kijken. Maar het is veel beter om alles op te splitsen in veel kleinere documenten. Op bijvoorbeeld 20 regels code of meer als het meer moet zijn maar ja, je hoeft het niet per se bij een ander document te doen. En als je dan dat bestand een mooie naam geeft, dan weet je altijd waar je moet kijken als je het zoekt		Werkte eerst niet met comments, nu soms.
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Niet. Omdat Grande Omega, daar laten ze een stuk weg en vul maar in. En ja dan leer je....ten eerste moet je aan het begin uit gaan zoeken, wat willen ze nou dat er gaat gebeuren. Maar dat is eigenlijk al heel lastig. En dan kan je in de zijkant.. dan is het mogelijk om terug te zien wat ze ongeveer van je willen. Vervolgens ga je kijken, hoe zou ik dat dan aanpassen. Dan moet je precies dezelfde benamingen gaan gebruiken als hun. Dat is dan ook niet het ergste, maar oké. Een vervolgens, als je alles goed hebt gedaan, dan is het opgelost. Maar je hebt daar geen vrijheid in. Je leert hun stukje code oplossen. Dat is programmeren niet.		Beschrijft stapsgewijs het verloop van de oefeningen in GO Vindt dat leren in GO bestaat uit het oplossen van code van GO.

Wat is programmeren wel?	Als iemand naar je toekomt en zegt: ik wil een login functie hebben. Je begint gewoon met een leeg iets en je gaat het maar maken. Dat is voor heel veel nieuwe programmeurs wel een hele grote stap en heel eng om zoiets leegs voor je te krijgen. Maar je moet ergens beginnen.	Ziet perspectief in ontwikkeling van GO. Heeft behoefte aan vrijheid in programmeren.
Heb je adviezen of aanbevelingen voor Grande Omega?	Ja. Het kan iets heel moois worden, maar momenteel is het dat nog niet, vind ik zelf. Het komt deels door de problemen die het heeft gehad met leerlingen er uitgooien, omdat het overbelast is. Dat is logisch, want je want je bent het nog aan het maken. Het is in ontwikkeling en dan horen er fouten in te zitten, maar niet zulke fouten. Op zich, de manier waarop Grande Omega werkt, als ik het goed heb begrepen, voor dat kleine stukje wat ik me erin verdiept heb, is dat ze wel je code echt na gaan lopen en ze gaan het een soort van uitvoeren en dat is iets heel moois, dat geeft toch deels vrijheid om iets te maken en vervolgens gaat het systeem het nalopen en kijken wat het doet en zo kan je mensen meer vrijheid geven. En dat dat zou super zijn, als ze dat lukt.	

Transcript interview 4 _onderzoek Grande Omega OP4 2018

Tijdstip: dinsdag 12 juni 2018

Plaats: HR Rotterdam

Interviewer: AvB

Geïnterviewde: 4, studentcode 32

Algemene vragen

vraag	**antwoord**	*kenmerken*
Wat is je vooropleiding?	Mijn vooropleiding was Havo, N&G, dat staat voor Natuur en Gezondheid. En daarbij heb ik ook als extra vak informatica gekozen. Daar had ik de keuze voor. Dus dat heb ik toen gedaan.	Voorkennis door keuzevak informatica op Havoniveau.
Omschrijf je programmeer-ervaring. Wat kon je al toen je begon aan dit collegejaar?	Ik kon HTML schrijven, een beetje. En ik had enige ervaring in PHP en enige ervaring in Java. Met enige ervaring bedoel ik: ik kon kleine programma's schrijven, maar ik begreep de concepten niet volledig. Zoals ik ze nu begrijp, zeg maar. Als ik PHP zou gebruiken, dan zou ik niet dingen kunnen doen zoals een rekenmachine maken met de kennis die ik toen had. Wat ik wel had gekund, is bijvoorbeeld een klein formuliertje maken, die je dan kon invullen. In Java was het meer toepasselijk. Er was een deel van Java dat we in HTML gebruikten, namelijk …een klein, klein ding was het, misschien om een animatie weer te geven of zo. Het kan ook dat het Javascript is, maar het is in ieder geval iets met Java. Dat gebruikten we om de animatie weer te geven op de website zelf.	Kon programma's (websites en klein project) maken maar begreep de concepten niet.
Heb je daarnaast misschien ook nog in je vrije tijd geprogrammeerd?	Uhm…niet echt geprogrammeerd, want ik had geen manier om dat te doen. Destijds, ik wist niks van de IDE's of wat dan ook. Het enige wat ik wist was de website die ik had gekregen van mijn school, destijds.	Van 4 naar 7.
Als je jezelf een cijfer mag geven van 1 t/m 10 voor je	Terugkijkend? Ja, nou, 4. Omdat ik vrij weinig wist in vergelijking met wat ik nu weet. Ik wist echt niet hoe weinig. Ik kon websites maken en ik kon een klein projectje maken	

programmeervaardigheden toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?	maar ik kon geen groot, uitbundig project maken. Meer was het opdrachten maken die met informatica te maken hadden, wat ik deed in dat vak.	
Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden na dit collegejaar, welk cijfer is dat dan? Om welke reden?	Uhm….een 7. Die verhoging zit in het feit dat ik meerdere talen op hoger.. - nou ja hoger - niveau kan toepassen dan dat ik eerst kon. Ik begrijp ook wat er precies gebeurt als je de code invoert. En het laatste is: ik kan grotere projecten maken omdat ik methodes heb geleerd om die projecten goed te kunnen uitvoeren.	30 tot 40 % GO, rest hoorcolleges DEV.
Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?	Hum..ik ga zeggen zo'n 30, 40%, daar tussenin. De rest aan hoorcolleges van Development.	

Themavragen Denken in Stappen

vraag	**antwoord**	*nv*	*kenmerken*
Beschrijf hoe je project 3 vanaf het begin aanpakte.	We hebben gewoon Scrum gedaan. We hebben toen 2 teams gemaakt, en ik zat met een iemand anders, vervolgens hebben onze product backlog gemaakt, elk team apart. Oh, die twee teams, ja we moesten dus een onderwerp krijgen en hoe de studie dat had opgesteld was: je hebt een groot team en dan zijn daar 5 mensen in dat past in je groep, zeg maar. En dan had je vragen, je had twee hoofdvragen en die moesten allebei in de applicatie beantwoord worden. In ons geval ging het dan om letsel en over inbraken. Ze waren gerelateerd, omdat de vraag daarboven was: hoe veilig kan je wonen? En toen we dat vastgesteld hadden, zijn we begonnen aan de methodiek van Scrum.		Beschrijft eerst een algemeen beeld van het project en noemt enkele losse componenten.
Dus jullie gingen die twee vragen verdelen over de teams?	Ja, dat moest. We hebben gebrainstormd door eerst te kijken wat is er allemaal beschikbaar was en vervolgens gebrainstormd over wat mensen überhaupt wel willen. Dat hadden we in een online call gedaan. En op die manier zijn we gaan kijken, nou iets wat een persoon wel zou willen tegenwoordig, is of ze veilig kunnen wonen en vervolgens was het, nou hoe gaan we dat dan weergeven. Want er zijn heel veel aspecten van, die veilig wonen zijn. Dus we hebben gekeken waar een goede dataset van beschikbaar was, die betrekkelijk was op dat….En vervolgens hebben we daar gekeken; welke vragen we kunnen opstellen met die informatie. Dus op die manier hebben we gekeken, nou we hebben letsel en we hebben overvallen en vervolgens hebben we dat zo opgesteld. Met de vraag hoe groot is de kans dat je letsel oploopt. Kans is een beetje moeilijk te berekenen maar we kunnen wel kijken hoe veel het is, in vergelijking met een andere gemeente. We hebben het per gemeente gedaan.	**H**	Beschrijft dan een specifieke, gestructureerde aanpak. Beschrijft gedetailleerd welke stappen zijn ondernomen.
En die vragen, voor	Die waren bestemd voor een gewone Nederlander, niet per se		

wie waren die bestemd?	een student, of iemand die een huis zoekt in ieder geval.		
Hoe hebben jullie dat aangepakt?	Heeft u het nu over de methodiek? Vanaf die vragen hebben we toen bedacht hoe we dat dan zouden weergeven. We hebben er een mock-up model van gemaakt, wat misschien een beetje ambitieus was, maar we hebben het wel gedaan. En daarin gaven we aan dat de gebruiker kan opzoeken in verschillende gemeentes of zelfs regio's en dat hij dan zo weergeeft hoe het werkt, en dat je ook nog een vergelijking functie hebt. Dat was de mock-up, de prototype van de applicatie. Dat hebben we met een computer gemaakte animatie gemaakt om het zo weer te geven aan onze product owner. Dus de persoon die het uiteindelijk moet goedkeuren.		
Hoe hield je het overzicht van wat je moest doen bij dit grote project?	We gebruikten Trello, dat is een programma waarmee je gemakkelijk kan weergeven welke taak iedereen heeft. Dus je maakt eerst de product backlog, daarin staan alle taken die uiteindelijk gedaan moeten worden. Dus aan wat voor dingen de product moet voldoen. En dan vervolgens gingen we door en dan gingen we een sprintplanning maken. Dat is een kleine periode, waarin je van alles moet doen. Om te zorgen dat je uiteindelijk een opleverbaar product hebt. En vervolgens hebben we dus die regels opgesteld waar een goeie lay-out voor bedacht moet zijn. Er moeten bepaalde functies in zitten et cetera. Per sprint zijn we zo gaan kijken. Wat kunnen we per sprint doen, zodat we een product hebben dat opleverbaar is en uiteindelijk hebben we een goed product gemaakt.	**H**	Beschrijft een specifieke, gestructureerde aanpak.
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en waarom je dat zo deed.	Reviews, ja die hebben we gehad, inderdaad. Aan het einde van de sprint, dat was inderdaad een sprintreview waarschijnlijk. En dan gaan we terugkijken wat er verkeerd is gegaan. En dat hadden we opgedeeld in verschillende plekken, dat was de dingen die niet goed gingen, dingen die misschien gevaarlijk zijn in de toekomst en als laatste de dingen die slecht zijn gegaan. Dit is even kortom geschetst. Ik weet de exacte namen niet, maar ik heb wel het document nog steeds. En dus daarin hebben we weergegeven van: wat is verkeerd gegaan. En toen hebben we dat toegepast op de volgende sprintplanning. Bijvoorbeeld als blijkt dat het een gevaar is dat we teveel inplannen omdat we daar misschien net uit zijn gekomen, dan kunnen we minder inplannen voor sprint 2.	**H**	Beschrijft een iteratie, vertelt het verloop en beargumenteert keuzes.
Op welke manier heb je een complex probleem zo geformuleerd dat het met code op te lossen was?	Ik neem aan dat het probleem van de vraag naar de code is? Nou, we zijn in ieder geval begonnen met een prototype maken, zoals ik al zei. En het prototype heeft ons geleid om na te denken over de functies die we dan in het programma moeten schrijven, want anders is het moeilijk om een prototype te maken, uiteraard. Zoals ik al zei, was het enigszins ambitieus maar het gaf wel goed weer wat de belangrijkste functionaliteiten zijn in het programma. Nou, natuurlijk om dat prototype op te stellen, hebben we wel een beetje nagedacht over hoe dat dan in zijn werk zou moeten gaan. En uiteindelijk zijn we dus met een prototype gekomen en toen konden we zo weergeven op een redelijk makkelijke manier welke functionaliteiten dan daar in voor moeten komen, terwijl we ook nog in gedachten houden hoe dat dan eruit zou moeten zien. Dat we niet vreemde functies erin zouden doen, die niemand zou begrijpen bijvoorbeeld.	**H**	Beschrijft een oplossings-richting via een prototype.
Je zegt, het was nogal ambitieus, wat gebeurde er	Nou, omdat we een animatie gebruikten, konden we letterlijk alles doen wat we wilden. Dus we hadden het per regio zoals ik al zei, dus we hadden het opgesplitst in OV-regio's en in die		Noemt niveaus van requirements.

dan?	OV-regio's, daar kon je dus op klikken en dan werd het zo uitgezoomd en dan kon je er weer op klikken en weer uitgezoomd maar de rest, de achterkant werd dan geblurred en dat soort dingen. Maar dat kan je natuurlijk moeilijk toepassen als een Must-have in het realistische project. Dus als je erop terug kijkt, was het te ambitieus maar je kon het wel als Could-have erbij zetten. Uiteindelijk is het zelfs zo geweest, dat het kaartgedeelte, dat we dat enigszins moesten verwaarlozen. Een subgroep heeft het niet gedaan, dat is de andere subgroep. Wij hebben dat wel gedaan, we hebben alleen weergegeven in welke provincie de gemeente zit.	**H**	
Bleken die pop-ups dan te veel werk?	Het bleek, omdat te doen, bleek het inderdaad teveel werk in vergelijking. En daarbij in gedachten houdend dat de rest ook nog gedaan moet worden. Want we hadden functionaliteiten om dingen te selecteren, om te vergelijken en om te zorgen dat je niet zomaar iets kan invullen wat helemaal nergens op slaat bijvoorbeeld. Dat soort dingen. En ja, op een gegeven ogenblik dan moet je dingen gaan schrappen. Als je zoiets ambitieus - terugkijkend is het ambitieus - gaat doen, dan moet je goed opletten.		Past strategie van aanpak aan.
Hoe maak je die keuze, wat je gaat schrappen?	Nou, we kijken naar dat deel…we gaan schrappen op de plek waarbij het nog steeds een applicatie is en de functie heeft die we willen dat hij heeft. Dus voor ons was het de bedoeling om weer te geven natuurlijk die vraag, dus hoeveel letsel is het dan bij ons en bij die andere was het: hoeveel overvallen. En uiteindelijk zijn we er dus achter gekomen, die animatie dat is niet een nodig iets om weer te geven, welke veiliger is of niet veiliger. Alleen meer waar het zit. Misschien handiger en gebruiksvriendelijker ook, maar het is niet een benodigdheid. Met die check boxes voor verschillende filters en een zoekbalk die automatisch bijvult naar welke gemeente je zoekt, hebben we het zo gedaan, zeg maar.		Maakt gebruik van schematische hulpmiddelen zoals tabellen met requirements.
Hoe wist je of de deliverable voldeed aan de vraag van het project?	Aan het begin van de periode ben ik persoonlijk naar de cursushandleiding gegaan en heb ik zo dat ding doorgespit. Uiteindelijk kwam ik dus met een tabelletje met alle requirements. In die requirements stonden bepaalde termen, termen die dus niet uitgelegd waren. Dingen zoals eenvoud, gebruiksvriendelijkheid of het moet antwoord geven op de vraag. Dus eenvoud, vraag, gebruiksvriendelijkheid. Die moeten goed gedefinieerd worden voordat we kunnen beginnen aan het project en dat is dus ook wat we gedaan hebben.		
Wanneer deden jullie dat, dat definiëren van die termen?	Nou we zijn gewoon met elkaar, een soort van, we hebben ernaar gekeken. Kort, eenvoudig is: dat er niet teveel dingen tegelijkertijd in je scherm zitten, dat je niet weet waar je moet beginnen. Eventueel zelfs als het gebruiksvriendelijkheid is, hetzelfde, maar er moet ook een mogelijkheid zijn voor jou als je niet begrijpt om het gewoon te begrijpen. Dat deden we in het begin van de fase, net voor de brainstormfase, omdat we dan bezig zijn met de cursus. Dus wat we uiteindelijk moeten gaan doen. En daarom zijn we begonnen met de vragen, zo ongeveer.		
Op welke wijze helpt het oefenen in Grande Omega bij de oplossingsrichting?	Voor development 3 is het een beetje een apart systeem. Want Grande Omega… wat we moesten is een database gebruiken. Ik moet eerst even illustreren welke functies we moesten hebben gebruikt in project 3, voordat ik kan dit kan vergelijken met Grande Omega. In project 3 moesten we veel gebruik maken van databases, omdat dit natuurlijk het hoofdonderwerp is van dat ding, van het hele project. En ook de dingen die in	Vindt geleerde in GO niet van toepassing op project. Ziet geen relatie tussen	

	WinForms gebruikt zijn. Maar het concept in Grande Omega, dat was destijds classes en functies en nog een aantal andere dingen. Die werden minimaal gebruikt in het project zelf. Vaak was het zelfs het geval dat we de code moesten opzoeken in plaats van het zelf moeten uitvinden in Grande Omega. Grande Omega, die helpt niet echt met het maken van welk project dan ook. Nou ja, zo is het.	GO en alle projecten.

Themavragen Abstraheren

vraag	**antwoord**	*nv*	*kenmerken*
Hoe pakte je je project 3 aan? En hoe organiseerde je bij project 3 de informatie die je nodig had om te kunnen programmeren?	Het maken van een database, het toepassen voor een bepaalde groep en… nadenken over hoe je iets weer gaat geven voor die groep.	**M**	Isoleert essentiële informatie
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?	Wat nu belangrijk is en niet belangrijk.. eh.. nou, we zijn dus met die instelling van wat belangrijk is en niet belangrijk gaan brainstormen. Uiteraard kwamen daar heel veel ideeën achter en die ideeën waren niet slecht. Maar, nu moet ik goed nadenken, want die dingen hebben we natuurlijk weggegooid en niet meer over nagedacht….	**M**	Kan relevante en minder relevante kwesties scheiden. Heeft eerder niveaus van requirements genoemd.
Hoe maak je dan de scheiding tussen: dit kunnen we weggooien en dit willen we echt houden?	Oké dat. De scheiding daartussen werd gemaakt, want we hadden de vragen opgesteld na de brainstorm. Naar de database, etc. We hebben gekeken naar hoe toepasselijk is het voor de gebruiker om dit idee te gebruiken, om dit toe te passen. Als je het toepast, hoe komt dat dan over voor zo'n doelgroep. Daar hebben we eerst gewoon over nagedacht. En uiteindelijk is het vaak zo dat de gekste ideeën, die worden er dan al uitgegooid. Volgens hadden we ook nog die vragen die we moesten beantwoorden. Op het moment van dat animatiegedoe, wilden we eerst die kaart. Dat is dus niet belangrijk geworden omdat we het niet kunnen toepassen. Dat was uiteindelijk een keuze. Maar in de beginfase dachten we er meer aan, hoe kunnen we het doen zodat het mogelijk is, niet super veel werk is maar dat het ook toepasselijk is en dat het ook op de cursushandleidingcriteria zit. Want als het teveel werk is dan moeten we teveel doen en dat is voor een student natuurlijk niet heel handig (lacht), want je hebt natuurlijk ok andere dingen te doen.		
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Grande Omega, welke onderdelen we doen of niet doen? Eh…niet. Analyse is meer daarvoor geschikt. Daarmee kijk je precies naar …Analyse 1 was dat, dat onderdeel: kijken naar of het project wat je gaat doen, of dat handig is, of het mogelijk is, et cetera. Dat werd behandeld in Analyse. Dus niet iets wat Grande Omega doet.	Ziet geen relatie tussen GO en het verduidelijken van de essentie. Wel relatie met Analyse.	

Themavragen Algoritmisch denken

vraag	**antwoord**	*nv*	*kenmerken*
Hoe zet je een probleem om in	Ik deel het eerst op in subproblemen, want vaak is een probleem heel groot. Zoals, bijvoorbeeld, wij moeten een app maken, nu	**H**	

code?	in project 4. Het probleem is, we hebben een grote app met een agenda en een plattegrond. We splitsen het op in verschillende problemen. Laat ik een voorbeeld nemen, de plattegrond. Als eerste moet je zeggen, nou, we moeten een plattegrond maken en waar gaan we dan beginnen want we moeten een basis hebben. Dus dan ga je onderzoek doen naar dat soort dingen. Waar je moet beginnen. En dan ga je kijken welke onderdelen van de plattegrond ga ik gebruiken, om te zorgen dat ik zo min mogelijk code moet schrijven, en zoveel mogelijk rendement heb uit die code. Dat is een moeilijke vraag, maar daar neem ik ook een redelijke tijd voor. Om te zorgen dat mijn code klein is. Want dan is het vaak ook - ik ga niet zeggen makkelijker- maar het is vaak wel handiger om dingen in op te zoeken. Want je weet meer en beter hoe je code eruitziet. Anders dan dat je 2000 lijnen gaat schrijven en dan moet je kijken precies waar het fout is gegaan, begin je op een plek, Oh, dat is verbonden met een andere plek, etcetera.		Benoemt specifieke stappen in logische volgorde en beschrijft de onderdelen. gedetailleerd (DIS).
Hoe maak je die code klein?	Ik maak eerst vaak een mock-up en dan is het is vrij groot en dan ga ik kijken hoe kan ik het doen zodat het simpeler is, maar dat het hetzelfde effect heeft. En vaak dan heeft het te maken met een ingebouwde functie, die dan niet per se met de taal te maken heeft, maar meer met het platform waarop je het gebruikt. Winforms bijvoorbeeld in project 3, die had ingebouwde functies om een kaart weer te geven dus wat we niet hoefden te doen is hele code te schrijven om een kaart te tekenen. Want die is al ingebouwd. Maar in de mock-up hebben we dus wel een afbeelding gebruikt. Dus je moet eerst laden voordat je kan beginnen aan dat soort dingen. En vervolgens heb je dan een kleine functie en dan als ik dat heb, dan heb ik een deel wat klaar is. Dan kan ik daar op op bouwen om meerdere delen te maken. Op dezelfde manier, maar het moet wel verbonden zijn met elkaar en omdat te krijgen, moet je goed nadenken over welke delen je gaat overnemen van het ene stuk. Want je hebt natuurlijk niet alles nodig. Bijvoorbeeld een plattegrond en een agenda. De app zelf, die gaat voor de hogeschool en dan kan je dus weergeven welk lokaal het is per agenda item. Dus dan moet je het verbinden met elkaar en dat is ook nog een vrij lastig systeem, dus dan moet je er misschien functies over maken, misschien redelijk groot schrijven en dan kan je het weer kleiner maken. Zodoende ga je dan door, totdat je alle subproblemen van het grotere probleem hebt opgelost.		Deelt grote problemen op in deelproblemen of deelvragen. Volgt een stap-voor stap benadering om proces uit te voeren. Hanteert een logische volgordelijk-heid.
Hoe ga je na of jouw code correct functioneert? (Als je merkt dat jouw code niet goed werkt, hoe ga je dan te werk?)	Oké, in de IDE's, ik heb er twee gebruikt in de tussentijd, eentje is Visual Studio en de andere is Android Studio. Beide hebben een ingebouwde functie, die heet breakpoint. En wat het programma doet, hij gaat elke keer naar elk lijntje van de code kijken. En met een breakpoint zeg je, dat lijntje code, daar stop je en dan kan je alle waarden bekijken en dan zeg je, ga nu naar de volgende stap en de volgende stap en de volgende stap en dan zo door. En kan je elke code bekijken waar het fout gaat. En dan kan je vaak - niet altijd - kijken wat er verkeerd is gegaan. Als ik het dan nog steeds niet weet, dan ga ik vragen stellen aan mensen die er meer verstand van hebben dan ik. Zoals bijvoorbeeld peercoaches.	**H**	Beschrijft een specifieke situatie en gebruikt tools. Denkt in stappen. Past zijn oude manier van coderen aan.
Heb je wel eens teruggekeken naar je oude code? Wat doe je nu anders?	Oja, (zucht en lacht). Het valt me op dat ik veel meer code nodig heb… gebruikt dan nodig was. Voor project 2 moesten we een game maken, dat hadden we dan in Python geschreven, dat is een andere taal. Maar het valt me op dat ik heel veel if-		

	statements gebruikt heb. If/else, if/else, if/else en dan heel het programma door geschreven, met die dingen. En het valt me op dat het veel kleiner had gemogen. While loop en…of misschien zelfs een functie geschreven zodat ik driekwart van de code niet elke keer opnieuw hoefde te schrijven. Wat wel gezegd is hoor trouwens, in development 2 en dat het misschien handig is, maar uiteindelijk heb ik gewoon…dacht ik er niet aan omdat ik dacht van; ik moet iets schrijven wat werkend is. Dat doe ik nu anders. Ik schrijf nu hoeveel werk het is en dan ga ik kijken hoe ik het kleiner kan maken. Wat ik bedoel daarmee, ik moet iets schrijven zodat het werkt en zodat ik goed kan zien wat alles doet en dan kan ik kijken hoe ik het kleiner kan maken. Want als je het groot schrijft en het dan uittest, dan kan je heel specifiek je breakpoint zetten en weet je precies waar het fout gaat. En dan je het kleiner maken, zodat het sneller draait bijvoorbeeld, of dat het simpeler is. Dat is het idee van code natuurlijk, dan moet je iets hebben wat het liefste één woord is en dat je dan een heel programma hebt.		Geeft specifiek aan wat zelf begrepen is en op welke wijze het beter kan.
Hoe zorg je ervoor dat jouw code begrijpelijk, werkbaar is voor anderen?	Wat ik vaak voor mezelf doe, omdat ik een beetje chaotisch ben soms, ik schrijf overal comments. Dus dan leg ik uit, dus ik - weer met plattegrond en agenda- dan maak ik een comment boven een deel van de plattegrond en dan zeg ik: dit is plattegrond, dit is waar ie gaat checken of het lokaal überhaupt bestaat. Et cetera. Dat zeg ik zo door. Want ik werk ook met andere mensen in mijn project, dus dan probeer ik dat zo te categoriseren, dat ik hier een stuk heb over de plattegrond en heb ik een klein stukje agenda en weer een plattegrond.	**M**	Noemt één kwaliteits-kenmerk en categoriseert informatie.
Op welke wijze helpt het oefenen in Grande Omega hierbij?	De manier waarop ik Grande Omega ervaar, is dat het je leert code van een ander te begrijpen. Dat is het meer. Het leert me niet hoe ik zelf code moet schrijven. Ofwel een zekere hoeveelheid, maar niet direct. Als ik een stukje code van Grande Omega lees, dan lees ik wat er staat en ik lees niet hoe ze het geschreven hebben of waarom ze het zo geschreven hebben, omdat het niet toepasselijk is op Grande Omega. Vaak in Grande Omega moet je alleen maar weergeven wat goed of fout is. In de multiple choice en in de forward en backward assignments moet je invullen wat er in de code mist, Het is dus niet zo dat je zelf de code schrijft, dus je hebt geen grondig idee van wat er in die code staat, verder. En toepassing van de dingen worden vaak alleen maar besproken in hoorcolleges van Development. Dat wordt niet echt gebruikt in Grande Omega. Zoals ik het zie is dat Grande Omega….die dient als een soort van poortwachter, zodat als je het niet begrijpt, dat je dan denkt van: oh wat? Ik heb het constant fout! En dan ga je kijken waarom je het fout hebt en dan oh, dat heb ik verkeerd begrepen…	Leert van GO de code van een ander te begrijpen, niet om zelf code te schrijven. Ziet GO als “poortwachter” om te controleren of code is begrepen.	
En als laatste, wil je nog iets kwijt over Grande Omega?	Ik heb het liever dat ik zie dat er meer feedback wordt gezet, want ik merk dat ik constant vastloop. Als ik iets fout heb, heb ik geen idee waarom ik het fout heb en dan moet ik naar een peercoach of een leraar, waardoor het dus niet echt zelfstandig is. En volgens mij was dat iets waar de HRO voor staat.	Mist inhoudelijke feedback van GO.	

Transcript interview 5 _onderzoek Grande Omega OP4 2018

Tijdstip: donderdag 14 juni 2018

Plaats: HR Rotterdam

Interviewer: AvB

Geïnterviewde: 5, studentcode 38

Algemene vragen

vraag	**antwoord**	*kenmerken*
Wat is je vooropleiding?	Ik heb een buitenlands diploma. Ik kom uit (..), daar heb ik scheikunde gestudeerd, drie jaar. Daarvoor had ik een havodiploma. Met behulp van dat diploma mag ik hier doorstuderen, hier bij hogeschool Rotterdam. Hiervoor moest ik het B2 niveau van Nederlands krijgen.	Geen voorkennis in programmeren.
Omschrijf je programmeer-ervaring. Wat kon je al toen je begon aan dit collegejaar?	Ik had geen enkele idee over programmeren. Maar in de beschrijving van de opleiding ging het over hoe leuk je het vindt om puzzels op te lossen, ja problemen op te lossen. Überhaupt moet je een paar uur achter een scherm zitten, bijna de helft van de dag achter een scherm zitten. Qua competenties dacht ik, ja dat zit in mij en daarom heb ik voor dit studie gekozen.	In vrije tijd geprobeerd een app te maken, niet voltooid.
Heb je daarnaast misschien ook nog in je vrije tijd geprogrammeerd?	Eigenlijk heb ik geprobeerd om een Android app te maken, voordat we hadden die op project, voor een antwoord uit te maken. Ja, maar toen krijg ik het heel druk met de studie en heb ik gestopt.	
Als je jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?	Tja, ik had geen helemaal geen ervaring dus ja, een 1.	Van 1 naar 4.
Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden na dit collegejaar, welk cijfer is dat dan? Om welke reden?	Ik geef 4. Qua hoorcolleges, die ik heb gekregen van meneer (..)., weet ik nu veel meer dan eerder over programmeren. Programmeerlogica eigenlijk. Met oefenen met Grande Omega, oefen ik de theorie. En ook tijdens het werken op de projecten krijg ik ook meer ervaring met het programmeren. Vooral kijk ik ook naar YouTube tutorials. Dat zijn handig om meer ervaring te werven.	65% GO, 20% projecten, 15% Youtube.
Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?	Eerlijk zeggen, het is 65% en de rest zijn, denk ik, 20% project en de rest komt van YouTube.	

Themavragen Denken in stappen

vraag	**antwoord**	*nv*	*kenmerken*
Beschrijf hoe je project 3 vanaf het begin aanpakte.	Bij project drie heb ik met een ervaren collega gewerkt. Ja, hij heeft al eerder zulke programma gedaan dus hij weet bijna alles over hoe moeten we die aanpakken. Hij wijst ons YouTube tutorial om ons te helpen. En ja, het was niet zo moeilijk voor mij want als ik ergens vastloop dan vraag ik aan hem. Toen kwam hij meteen met een oplossing. Desondanks heb ik aan het eind van de project een probleem met zijn code opgelost. Hij was vastgelopen en ik was ontwikkeld aan het einde van de project, dus heb ik de probleem voor hem opgelost. Het was een probleem met de logica. En door het hoorcollege, delen snap ik wel. Daarom had ik de oplossing. Het was eigenlijk een probleem met de nieuwe combobox. Dus een combobox om menu te krijgen. Binnen onze app was er vier combobox. Wij deden voor een van de diagrammen.	**H**	Beschrijft gedetailleerde aanpak in stappen. Heeft een logicaprobleem opgelost voor een ervaren collega.
Kan je iets vertellen over hoe je dat probleem hebt opgelost?	Ik ga voor de eerste een land kiezen dan verschijnt het in de derde combobox dus er was een connectie tussen dezen en dat mag niet eigenlijk. En ja, heb ik ze uit elkaar gemaakt. De combobox, is de menu met een kleine pijltje. Als je op die pijltje drukt, dan krijg je de hele menu.		Legt uit wat de logica voorschrijft.
Hoe heb je dat gedaan?	Eigenlijk heb ik voor elk combobox een aparte code geschreven. Dat waren dezelfde code en daarom verschijnt ze op dezelfde manier.		
Hoe wist je dat?	Omdat de logica zegt, dat moet niet dezelfde combobox zijn, ze moeten iets anders geven. Daarom gaan ze niet op dezelfde code zijn. Ik heb de code gelezen. De uitleg van de logica, die ik heb in de hoorcollege..die wist ik wel, qua logica, dat mag niet.		
Hoe hield je het overzicht van wat je moest doen bij dit grote project?	Eigenlijk, toen we begonnen met werken, was het idee om te baseren, om een website voor gamen. En we kunnen de data niet eruit halen daarom zijn we voor een andere gaan kiezen en toen kwam ik met een idee om de modale inkomen van een Europees land te vergelijken met de hoeveelheid van dit modale inkomen wordt besteden aan de online handel. Eigenlijk dit was de het idee voor onze subgroep. De andere groepen waren blijven met die website van die gamers en toen moeten ze geen echte data van de website hebben, ze moeten de database zelf aanmaken, Maar ik heb de database voor de hele project gemaakt. Dat was eigenlijk mijn eerste taak.	**M**	Beschrijft een probleem en een aanpak voor de oplossing.
Hoe hield jij nou bij, of hoe checkte jij, of je aan het eind met een goede oplossing kwam?	Voor het probleem? Toen de app geeft de juiste visualisatie van de data, toen het was duidelijk, het verschil tussen de modale inkomen en de mate van online handel. Ik merkte, de app werkt wel. Het is duidelijk: de verschillen tussen de modale inkomen en de verkopen.		
Op welke manier heb je een complex probleem zo geformuleerd dat het met code op te lossen was?	Met code? Dan moet ik eerst de data binnen mijn database brengen en dat doe ik met een SQL-query. En dan moet ik die ook zelf oproepen, ook met de SQL-query's. En dan met de C# code, die veranderen naar schema's en dat heeft C# in Visual Studio. Dat was niet zo moeilijk om te converteren.	**H**	Beschrijft een specifieke, gestructureerde aanpak.
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en waarom je dat zo deed.	Ja, elke sprint we hadden een retrospective, maar we hadden eigenlijk niet de proces voor programmeren zelf. Ja, voor samenwerking, wat ging goed, wat ging niet goed. We voorstellen een oplossing voor dit en we passen dit toe op de volgende sprint.	**H**	Maakt onderscheid in revisie van proces en van code.

Op welke wijze helpt het oefenen in Grande Omega bij de oplossingsrichting?	Het helpt om te oefenen op de logica. Maar eigenlijk, het heeft een andere syntax, dus het was niet zo handig om te wisselen tussen de syntax van Grande Omega en C#. Soms weet ik al de oplossing voor de probleem, maar moet ik de syntax opzoeken.	Ziet relatie GO en logica.

Themavragen Abstraheren

vraag	**antwoord**	*nv*	*kenmerken*
Hoe zorgde je ervoor dat je wist wanneer welke sprint was en wat je moest doen?	Met een sprintplanning. Voor elke sprint stellen we op een sprintplanning met use cases, de taken die moeten uitgevoerd worden en ook met de acceptatiecriteria voor de user story's. En we verdelen de taken met Trello. Op Trello, iemand krijgt zijn eigen taak en plaatst zijn naam op die taak. Dus dit werkt, dit wordt niet tweemaal gedaan.	**H**	Beschrijft specifieke aanpak met schematische hulpmiddelen. Vindt dat leerlijn Analyse bij projecten helpt.
Op welke wijze helpt Grande Omega bij het organiseren van je project?	Grande Omega staat niet voor organiseren van je project. Analyse helpt meer met organiseren van een project, want daar staan de diagrammen voor de classes, voor de state machine, de activity diagrams.		
Wat waren de belangrijkste onderdelen van het project?	De data. De data vervangen van de website en de type van de databestanden en die binnen de database verwerken.	**H**	Kan relevante en minder relevante kwesties scheiden. Herkent eisen en prioriteert eisen.
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?	Hoe belangrijk die ding is voor de project. Bijvoorbeeld, de lay-out is niet het belangrijkste in het project, want het geeft wel een uitstraling voor de app maar als de app werkt niet soepel, dan maakt de lay-out niet uit. Dan heeft het geen nut om veel tijd te besteden aan de lay-out. De app is dan al klaar.		
Hoe maakte je nog meer onderscheid?	De productbacklog, de functionaliteiten die moeten komen in de eindproduct en daarin zetten we de prioriteit voor elke functionaliteit. Hoe belangrijk is de functionaliteit, hoe belangrijk is de code en de werken aan de functionaliteit.		
En hoe stelden jullie die prioriteit?	Qua de probleem. Sommige functionaliteiten zijn heel belangrijk voor de probleem en de rest misschien zijn ze qua gebruikersgemak belangrijk. En dan de eisen, die gaan eigenlijk de echte problemen oplossen, die zijn de belangrijkste.		
Hoe maakte je onderscheid in al die verschillende eisen?	Eh…Eigenlijk alle functionaliteiten die zijn belangrijk voor de probleem. Dezelfde mate, dus we gaan geen onderscheid ertussen zetten.		
Op welke wijze helpt het oefenen in Grande Omega hierbij? Bij het onderscheid maken tussen wat belangrijk en niet belangrijk is?	Volgens mij heeft Grande Omega niet te maken met onderscheiden binnen de projecten, want Grande Omega is meer gebaseerd op de logica van de programma. Hoe moet het, de logica van de programmeertalen. We zijn nog bezig met de basis van programmeertalen. Soms staat een groot verschil tussen de mate van de Grande Omega en de code die we moeten voor de app verwerken. Dus we krijgen de classes binnen de programmeertalen en de derde periode moet ik Grande Omega doen en we waren al bezig met de classes binnen die de eerste, de tweede en de derde project. En ik denk dat, de betekenis is niet met de projecten. En nu wordt beloofd dat de volgende jaren zou niet zo zijn. Volgende jaren is dat een project voor twee perioden zijn. En dan denk ik, het wordt beter. Dan kunnen de docenten van Grande Omega met		Ziet geen relatie tussen GO en het verduidelijken van de essentie. Ziet een groot verschil tussen wat geleerd wordt in GO en wat in het project moet worden toegepast.

	die project ook echt helpen.	
Dus eigenlijk zeg je: we moesten al werken op een manier die we nog niet hadden geleerd?	Ja.	

Themavragen Algoritmisch denken

vraag	**antwoord**	*nv*	*kenmerken*
Hoe zet je een probleem om in code?	Het hangt af van de vraag, eigenlijk. Is het een vraag qua logica van de code, dan kan ik misschien iets bedenken en dit te proberen met een programmeertool, zoals Visual Studio. Maar als het ging over een syntax of zoiets, dan moet ik naar Google en Stackflow. Daar kan ik de syntax vinden. Meestal zijn de syntax een voorbeeld van de logica. Dan probeer ik het op mijn app, past het op mijn probleem. Dan schrijf ik de code binnen mijn app, binnen de code van mijn app en ik run het en dan krijg ik het resultaat en dan zie ik waar het ging niet goed. En dan kan ik dit veranderen. Dus altijd als ik werk met coderen, als ik denk dat: ik heb het, dan run ik. Aan de hand van de resultaten kan ik weten van wat ging goed, wat niet goed.	**H**	Beschrijft specifieke stapsgewijze aanpak van probleem, controleert en reflecteert.
Hoe ga je na of jouw code correct functioneert? (Als je merkt dat jouw code niet goed werkt, hoe ga je dan te werk?)	Als het ging goed, dan werk ik, ik kan het mooier maken. De code netjes doen, dus comments voor elke stukje code schrijven. Soms schrijf ik iets, die is niet nodig voor de oplossing, dan haal ik die eraf. Of maak ik een comment ervan. De meeste van de coderingtools geven een soort van rode lijn onder de foute code en dan kan ik opzoeken. Meestal zoek ik het op internet op, Google. En dan komt dit met dezelfde situatie, die de meeste programma's runnen, tegen.	**H**	Beschrijft een specifieke situatie en controleert de oplossing.
Zijn er nog meer manieren, als je code niet goed werkt, om te zorgen dat je weet wat er aan de hand is?	Ja, de code checken. De code qua logica checken. Misschien zit er iets fout met de code. Of soms dit is een fout met de Syntax. Soms, een kleine komma kan een groot verschil maken.		Werkte eerst nooit met comments, nu vaak.
Dat doe je dan altijd door zo'n codering tool te gebruiken?	Ja. Grande Omega helpt wel met code lezen. Want de backward assignments in Grande Omega zijn gebaseerd op het lezen van de code en de foutjes invullen. Of de meeste dingen invullen. Ik heb geleerd om precies te kijken naar elk klein detail vanuit de code.		
Heb je wel eens teruggekeken naar je oude code? Begreep je die toen nog?	Eigenlijk niet (lacht). Ik kijk niet terug naar mijn oude code, want ik denk: ze zijn niet zo relevant voor nu. Voor mij. In het tweede project heb ik met Python gewerkt en het derde periode heb ik C# en nu werk ik met Java. Het zijn verschillende talen dus ja, ik denk, ik ga niet iets pakken van de andere programma's die van nut zijn voor mij.		

Wat doe je nu anders? Doe je nu iets anders wat betreft het schrijven van je code als in het begin? Zit er voor jouw gevoel ontwikkeling in jouw manier van code schrijven?	Ja, in de tweede project schrijf ik nooit een comment. Ik schrijf altijd code zonder comments. Ja, dat was rommelig. In het begin schreef ik het meeste met hard codering, met if-statement. Dat gebruik ik veel. Nu werk ik meer met for loops en while loops en meer arrays in plaats van altijd…dus in een array en na de array een if-statement. Dus dat scheelt ook op de tijd. Dan geldt het voor de hele array, in plaats van een if-statement voor elk element.		
Hoe zorg je ervoor dat jouw code begrijpelijk en werkbaar is voor anderen?	Comments. Comments voor de code. En ook de benaming van de classes en functies, dat helpt ook. Het moet zo duidelijk zijn: wat is de bedoeling van dit functie of dit class.	**H**	Noemt drie kwaliteits-kenmerken
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Hoe ga ik met de arrays om, hoe ga ik met de if-statement om en de while loops en for loops, de logica van dit functionaliteit heb ik allemaal geleerd van Grande Omega. Eigenlijk precies door de hoorcolleges en geoefend met Grande Omega.	Heeft in GO vooral de logica van de functionaliteiten geleerd.	
En als laatste, wil je nog iets kwijt over Grande Omega?	Ik denk dat het is beter als Grande Omega gebruikt dezelfde syntax van de echte talen. Ik weet het wel, dat het is een doel voor …om de logica te leren, maar die verschil tussen die syntax van Grande Omega en de echte bestaande talen maakt het soms verwarrend voor de studenten. Grande Omega gebruikt dan bijvoorbeeld de syntax van C#, want de meeste studenten werken met C#. Dan wordt het duidelijker voor de studenten. Ze kunnen dezelfde zien als ze gaan werken aan het project. Niet dezelfde logica, maar met een andere syntax.	Vindt verschil tussen de syntax van GO en C# verwarrend.	

Transcript interview 6 _onderzoek Grande Omega OP4 2018

Tijdstip: woensdag 20 juni 2018

Plaats: HR Rotterdam

Interviewer: AvB

Geïnterviewde: 6, studentcode 65

Algemene vragen

vraag	**antwoord**	*Kernpunten*
Wat is je vooropleiding?	Havo.	Voorkennis in programmeren door vooropleiding en functie als ICT'er bij klein bedrijf.
Omschrijf je programmeer-ervaring. Wat kon je	Ik had ervaring in PHP, HTML en CSS. Want we hadden bij ons op school ook het vak informatica. Dat was het wel grotendeels, wat we daar leerden. Een keuzevak, ja.	

al toen je begon aan dit collegejaar?		
Wat deed je nog meer?	Af en toe een beetje zelf, maar dat was meestal ook PHP of HTML.	Ervaring in het ontwikkelen van programma's.
Als je jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?	Mwah…een zeventje. Nou…op mijn havo-school kreeg ik best wel hoge cijfers voor informatica. En we hadden daar ook projecten enzo...en ik kreeg daar toentertijd alternatieve opdrachten voor omdat andere opdrachten te makkelijk waren voor mij.	Van 7 naar 8,4. 15% GO en 85% project en zelfstudie.
Je dacht: ik heb meer uitdaging nodig? Hoe kreeg je die uitdaging?	Nou de leraar kwam gewoon naar mij toe en dan… ja, jij krijgt een alternatieve opdracht. De laatste opdracht op die school was een systeem maken via een website die…waar mee je kon zien wat je voor je laatste cijfers moest halen om een voldoende gemiddeld te staan. Dan vulde je je cijfers in en berekende hij automatisch wat je de rest van het jaar moest halen. Dat was gebouwd in HTML en PHP. Compiling had ik nog niet gedaan, dat was niet nodig.	Leert bij GO het begrijpen van code. Leert bij projecten coderen en taal.
Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden na dit collegejaar, welk cijfer is dat dan? Om welke reden?	Uh…8,4 denk ik. Meerdere talen geleerd en het is nu zelfs m'n werk geworden dus…Ik ben een ICT-medewerker sinds anderhalve maand denk ik, en ja dat gaat goed. Bij (noemt bedrijf). Wij zijn een bedrijf die werken voor bijvoorbeeld (energiebedrijf), die langs je deur komen en dan wil je overstappen en dan hebben ze een tablet en dan hebben ze daar een applicatie voor. Of we werken voor goeie doelen. Als u een keer geld wilt overmaken voor een goed doel, dan gaat dat allemaal via een programma dat wij hebben gemaakt. En problemen daarin lossen wij ook op.	
Ben je dan junior developer?	Nee, echt een ICT-medewerker. We zijn met z'n zessen, niet zo'n heel groot bedrijf eigenlijk. Wel veel klanten.	
Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?	Ja… ik denk..15% aan Grande Omega. Ja, en de rest is echt project en zelfstudie. Grande Omega leert je, zeg maar, het begrijpen van code, daar is het vooral voor bedoeld. Maar code zelf begrijp ik grotendeels al. Dus ik leer er niet heel veel uit.	
Wat heb je dan vooral bij de projecten geleerd?	Het coderen. De taal en dat soort dingen.	

Themavragen Denken in stappen

vraag	**antwoord**	*nv*	*Kernpunten*
Beschrijf hoe je project 3 vanaf het begin aanpakte.	In het begin gingen we al redelijk rap. We gingen natuurlijk eerst gewoon een onderwerp zoeken. Al vrij snel in het begin gingen we een applicatie maken.	**H**	Beschrijft een combinatie van agile methode incl. schematische hulpmiddelen en algemene voorbeelden van poging tot
Daartussen gebeurde ook nog wat…	Ja, we gingen nog kijken wat de belangrijkste punten waren zeg maar, via Scrum. En user back log en project back log opschrijven en dat soort dingen. En vanuit daar zijn we langzaam alles gaan uitwerken.		
Hoe ben je dat gaan doen?	Nou ja, vooral op Internet kijken hoe je bepaalde dingen moest doen. Natuurlijk het begin, dat ging al vrij snel, want dat was		

	gewoon basiscoderen, dat lukt wel. Maar uiteindelijk kwam je natuurlijk wel op bepaalde functies waar je vastliep. Dat was gewoon opzoeken op Internet.		structuur.
Kan je een voorbeeld geven?	We moesten een…hoe heet het nou…een heatmap maken. En die moest reageren op de waarden die je had ingevuld. En ja, dat ging niet in een keer. Op Internet staan genoeg voorbeelden voor heat maps dus eh…daar hebben we wel delen uitgehaald en gecombineerd en zo. Uiteindelijk werkte het.		
Hoe hield je het overzicht over dat hele project?	Met Trello. Trello is zeg maar een scrumboard.	**H**	Geeft specifieke voorbeelden
Wat deden jullie, hoe ging dat?	We hadden dan bepaalde sprints. En in die sprint namen we een paar dingen op van de backlog die we gingen uitwerken in twee weken. En daar koppelden we gewoon een persoon aan.		
Jullie koppelden een persoon aan zaken die in de product backlog stonden. Hoe ging dat verder?	Nou, die gingen dat dan uitwerken en op het moment dat je eerder klaar was dan de rest, dan ging je gewoon de rest helpen eigenlijk. Want niet iedereen in je groepje kan even snel of even goed programmeren en sommige functies zijn gewoon makkelijker dan andere dus….		
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en waarom je dat zo deed.	We gingen gewoon met de product owner zitten en we lieten zien wat we tot nu toe hadden. En hij vroeg wat we twee weken daarna zouden opleveren.	**H**	Maakt onderscheid in Scrum-reviews en feedback van de product owner. Noemt kwaliteitseisen van op te leveren product
Hoe wist je wat je twee weken daarna moest opleveren?	Dat was een beetje overleggen met de product owner, wat haalbaar was, voor ons idee en daar is uiteindelijk wel wat uitgekomen dan. Voor Scrum hadden we drie reviews en voor de product owner ook drie of vier, zoiets.		
Was het altijd hetzelfde, gebeurde er ook wel eens iets anders?	Nou, de eerste product owner meeting is natuurlijk dat je voor gaat leggen wat je gaat maken. En misschien extra functies daarvan. Eigenlijk het hele project, beetje samenvatten. En ja, uiteindelijk, wat ga je doen in twee weken. Maar de eerste is dan net iets anders.		
Hoe wist je wat je aan het eind moest opleveren?	Ja, dat is meestal gewoon een werkende applicatie. Iets wat werkt. De opdracht, je had gewoon een opdrachtblaadje, waarin stond wat de minimumeisen waren om in ieder geval het project te halen. En daar ga je natuurlijk als eerste achteraan. Dingen zoals: de applicatie moet geen bugs bevatten, moet niet crashen, dat soort dingen. We moesten voor het project ook gebruik maken van CSV bestanden, databestanden….Dat was het wel denk ik, dat waren een beetje de grote eisen.		
Hoe ben je met het omvangrijke probleem van project 3 omgegaan, zodat je het met code kon oplossen?	Ja, we gingen eerst kijken naar de CSV's. Wat daar nou de makkelijkste probleemstelling voor was. En dat bleek bij ons dus te zijn, waar je het beste kan wonen in Utrecht. En dat is opgelost door de CSV bestanden te gebruiken en dat in een heatmap te zetten. En op het moment dat de kaart dan een deel groen is, dan is het daar goed om te wonen. En een deel rood dan is het slechter.	**H**	Onderzoekt probleem-stelling en hanteert volgordelijkheid
Wat betekent de afkorting CSV?	Geen idee, CSV is eigenlijk gewoon een bestandje met data erin. Het is een bestandje met bijvoorbeeld misdrijven en dan in de regio van Utrecht en dan hoeveel misdrijven er zijn in een week, in een maand, in een jaar…		
Op welke wijze helpt het oefenen in Grande Omega bij het oplossen van zo'n probleem?	Eh…ja…geen idee eigenlijk. Grande Omega heeft niet echt heel erg meegeholpen in het project.	Ziet geen relatie tussen GO en project.	

Themavragen Abstraheren

vraag	**antwoord**	*nv*	*Kernpunten*
Wat waren de belangrijkste onderdelen van project 3?	Het gebruiken van een CSV-bestand. Dus data en ja, je hoofdvraag moest opgelost worden. Dat waren wel de belangrijkste.	**H**	Kan de essentiële informatie isoleren.
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?	We hadden in ons Trello-board kleurtjes gecombineerd met Must, Could en Should. Must was dan rood, het meest belangrijk, zeg maar de hoofdfuncties, Should was de dingen die zouden moeten, dat is…ze zijn wel belangrijk maar ze hoeven niet per se. De Could's zijn de extra's, dat zijn bonuspuntjes, zeg maar.	**H**	Geeft drie niveaus van requirements aan en legt deze uit.
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Ja, niet echt eigenlijk. Bij Grande Omega heb je niet echt Must's, Should's en Could's. Je hebt gewoon opdrachten. Nou zijn die allemaal wel redelijk belangrijk voor de toets, uiteindelijk. De toets gaat echt wel volledig over Grande Omega.	Ziet geen relatie tussen GO en proces van essentie verduidelijken.	

Themavragen Algoritmisch denken

vraag	**antwoord**	*nv*	*Kernpunten*
Hoe zet jij een probleem om in code?	Opzoeken op Internet, hoe ik dat moet doen meestal. Je hebt genoeg websites, documentaties van bepaalde code of eh…hoe heet die website nou…je hebt een website waar iedereen kan vragen over code en dat er gewoon geantwoord wordt door anderen.	**H**	Beschrijft zoekstrategie via Google en noemt forum. Omschrijft logisch nadenken als een volgordelijke handeling.
Is dat Stack Overflow?	Ja, Stack Overflow! Dat staat ook meestal bovenaan Google op het moment dat je naar een probleem zoekt.		
Kom je altijd tot een oplossing?	Nee, niet altijd. Maar wel een deel van het probleem, kan je dan oplossen daarmee. En zo raap je alles, zeg maar, een beetje bij elkaar.		
De rest heb je dan nog niet. Wat doe je dan?	Verder zoeken. Ook weer naar Google. Soms lukt het ook wel om het zelf gewoon te proberen. En dan een beetje logisch nadenken, dat lukt het ook wel eens.		
Wat houdt dat voor jou in, logisch nadenken?	Bijvoorbeeld, als je ergens een for loop voor nodig hebt, dat je dan gewoon getalletjes aanmaakt voor die for loop zodat ie altijd blijft lopen, tot zo vaak.		
Hoe ga je na of jouw code correct functioneert? (Als je merkt dat jouw code niet goed werkt, hoe ga je dan te werk?)	Je gaat 'm sowieso eerst testen. Dus gewoon het programma aanzetten. Dat kan meestal wel gelijk. En mocht ie crashen of foutmeldingen geven, dan krijg je die foutmeldingen ook meestal volledig uitgeschreven. En dan kan je die foutmeldingen ook gewoon googelen. Of het is gewoon iets simpels zoals een punt komma, of een haakje missen.	**H**	Beschrijft via spcifieke stappen een aanpak en controleert of deze werkt.

Hoe test je dan? Heb je nog andere voorbeelden?	Door gewoon het programma aan te zetten. Als je bijvoorbeeld de heat map wil testen dan vul je waarden in en dan druk je dus op confirm en dan hoort als het goed is de heat map te veranderen. Dan maakt het niet uit of de gegevens correct zijn maar je weet in ieder geval dat die functie al werkt. In Op4 hebben we nu een app en die blijf je gewoon testen door de hele tijd op je telefoon aan te zetten en dan crasht ie en dan kijk je opnieuw naar je code, en weer aanzetten…		
Hoe zorg je ervoor dat jouw code ook werkbaar is voor anderen?	Daar gebruiken we GitHub voor. GithHub is een platform voor eigenlijk…coders. GitHUb zorgt ervoor dat de codes worden samengevoegd in de bestanden,zeg maar.	**M**	Noemt geen kwaliteits-kenmerken, wel platform voor coders.
Wat voor bestanden zijn dat dan?	Meestal, ja, ligt eraan wat voor code je typt, maar Javascript, C#...		
Heb je wel eens teruggekeken naar je oude code? Wat doe je nu anders?	Nee, ik heb niet teruggekeken naar mijn oude code. Die heb ik ook niet meer. Van mijn Havo-school, dat stond daar allemaal op de computer in het systeem. Dat heb ik nooit thuis gehad. Ik kan nu natuurlijk wel kijken naar de code die ik heb geschreven in m'n opleiding. Bepaalde functies daarvan kan je natuurlijk ook gebruiken in andere…projecten. Dat is gewoon kopiëren en plakken eigenlijk. Soms een paar kleine aanpassingen zodat het wel werkt. Variabelen, uitkomsten, dat grotendeels.	**M**	Beschrijft geen verbeterpunten maar denkt wel na over toekomstig aandachtspunt.
Pak je die variabelen anders aan dan in het begin van het jaar of is dat hetzelfde?	Neu… dat is meestal wel hetzelfde.		
Op welke wijze helpt het oefenen in Grande Omega jou hierbij?	Eh...ja…uhm.. je leert met Grande Omega wel wat de uitkomst van zo'n probleem, kan zijn. Zo'n functie.. En…daar blijft het wel bij, denk ik.	Leert van GO wat de uitkomst van een codeprobleem kan zijn. Heeft behoefte aan debugger met inhoudelijke feedback.	
Wat je leert, hoe kan je dat zelf gebruiken als programmeur?	Nou, bijvoorbeeld als je functies schrijft en daar moet true of false uitkomen, dan kan je dat natuurlijk ook gewoon zelf in je code neerzetten.		
Heb je dat weleens gedaan?	Een true of false statement sowieso.		
Was dat nieuw voor jou?	Nee, dat had ik al met PHP gedaan.		
Wil je nog iets kwijt over Grande Omega? Nog aanbevelingen?	Een goeie debugger. Aangeven wat je fout hebt gedaan in je code. Want ze hebben wel een deel-debugger en soms geeft die wel goed aan wat je fout hebt gedaan, maar soms geeft ie iets aan waar je niks mee kan.		
Wat gebeurt er dan, als je er niets mee kan? Wat doe jij ermee?	De opdracht opnieuw proberen. De error die hij dan geeft, geeft geen verklaring. Of waar. Ja, dan staat er wel waar, maar dat is eigenlijk in een back-end script. Dus daar kunnen we sowieso niet bij.		

Transcript interview 7 _onderzoek Grande Omega OP4 2018

Tijdstip: woensdag 20 juni 2018

Plaats: HR Rotterdam

Interviewer: AvB

Geïnterviewde: 7, studentcode 68

Algemene vragen

vraag	**antwoord**	*Kernpunten*
Wat is je vooropleiding?	Hiervoor heb ik mbo-4 gedaan, middenkader engineering.	Geen voorkennis in programmeren
Omschrijf je programmeer-ervaring. Wat kon je al toen je begon aan dit collegejaar?	Eh…helemaal niks, zou je gerust kunnen zeggen. Ja, wel een beetje, misschien in bepaalde bestanden van spelletjes induiken maar niet echt concreet Java of wat dan ook.	Geen ervaring in het ontwikkelen van programma's.
Wat ging je doen in die bestanden van spelletjes?	Eh gewoon, dat zijn Mods, dat zijn aanpassingen maken en dat ga je verfijnen naar je eigen smaak.	
Als je jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?	Ja, 0. Ik had geen ervaring in alle talen, helemaal niks.	Van 0 naar 2
Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden na dit collegejaar, welk cijfer is dat dan? Om welke reden?	Ja…het blijft natuurlijk mijn eigen beoordeling…dan zou ik tegen mezelf zeggen dat ik nu, ja…er is nog zoveel te leren en zoveel te doen, dus het zou een beetje raar zijn om mezelf een hoog punt te geven, dus ik zeg…ik denk een 2 of zo. Ik wil mezelf nog geen voldoende geven, nee.	Vindt dat beheersing van programmeer-niveau nog onvoldoende is.
Waarom geef je jezelf deze 2?	Omdat ik merk dat ik met programmeren, en ook met projecten en dergelijke, dat ik wat te veel bezig ben met basisdingen onder de knie te krijgen, van het programmeren zelf. Dus wat bepaalde woorden betekenen, bepaalde functies en dergelijke. Een daarop is dat gebaseerd. Dat ik van mezelf vind, die basis is er nog niet.	
Als je kijkt naar jouw resultaten bij Analyse en Dev, waar sta je dan ongeveer gemiddeld?	Ik sta nu voldoende. Analyse wat beter maar voor Dev sta ik tot nu toe voldoende, ruim voldoende.	60% GO 40% Youtube tutorials en klasgenoten, ook copy-paste acties.
Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?	Dan zou ik zeggen, ik denk 60% Grande Omega. En 40% filmpjes kijken op YouTube. Veel klasgenootjes vragen. Kopiëren en plakken.	Ziet relatie tussen

Wat heb je geleerd in Grande Omega?	Dan heb ik het met name over het eerste en tweede blok, zeg maar. Daar merkte ik wel, dat ging best wel goed, zeg maar. Hoe meer tijd ik erin stopte, hoe beter ik het ging begrijpen. Maar de sprong van twee naar drie, dat was toch wel andere koek, op z'n zachtst gezegd. Ik merkte, ik kreeg niet dezelfde voldoening als bij de eerste en de tweede. Bij de eerste en tweede had ik echt het gevoel, oké, ik begrijp het en ik ben iets verder. Maar vanaf de derde, als ik ergens tegenaan botste, dan bood het programma niet echt de oplossing voor mij. Dat ik weer effe verder kon.	inspanning en leerresultaat. Sprong naar OP3 was groot. GO hielp daar niet bij.

Themavragen Denken in stappen

vraag	**antwoord**	*nv*	*kernpunten*
Beschrijf hoe je project 3 vanaf het begin aanpakte.	Dat eh…weet je wat het is, vanaf project drie moet je gewoon geluk hebben dat je klasgenoten hebt die programmeerervaring hebben. Als je iemand hebt in je team die wat programmeerervaring heeft, dan zit je meestal goed. Want die sleept de hele groep eruit en weet je hoe het allemaal werkt, een beetje. Maar als je dat niet hebt, dan wordt het vrij lastig.	**L**	Rudimentaire beschrijving van aanpak. Geen methodische of systematische aanpak. Probleem wordt door een ander in het team opgedeeld of door anderen zodanig geformuleerd dat het m.b.v. een computer is op te lossen.
En hadden jullie die?	Die hadden we, ja.		
Hoe pakten jullie dat samen aan?	In ons geval dat diegene met programmeerervaring een platform opzet, waar iedereen dus in kan en dat zorgt ervoor dat diegene ook kan controleren, wat wordt aangepast en wat niet. En iedereen kreeg zijn onderdeel, gedeelte waar hij aan moest werken en ja, dat is het eigenlijk.		
Hoe pakte jij dat zelf aan?	Ik kreeg ook een onderdeel en die deed ik ook nog samen met iemand die ook wat verder was dan mij. En wat je dan krijgt is dat die…die er veel meer verstand van heeft, die ragt er snel doorheen, waardoor je jezelf niet echt ontwikkelt omdat je denkt, oh, het is al gedaan. Mooi. Ik kan bezig zijn met Scrumdocumenten ofzo. Dus dan ga je het anders invullen. Dan ben je niet echt bezig met code schrijven of wat dan ook.		
Hoe hield je het overzicht over wat je moest doen?	Dat was gewoon op Trello kijken via de app, jongens wat moet er gebeuren. Via Whatsapp, niets bijzonders. Ja, zo hadden we contact met elkaar, ook gewoon face to face.		
Hoe ging dat dan op het scrum board, opTrello?	Wat zal ik zeggen. Je gaat voor de computer zitten en je gaat, wat staat er op Trello en wat moet ik gaan doen.		
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en waarom je dat zo deed.	Wat wij deden was…dat reviewgebeuren deden we niet bij elkaar maar deden we op OneDrive met elkaar. Op OneDrive hadden we bestanden gezet en bij een retrospective, dan gingen we voor zichzelf, thuis, invullen wat ie van de ander vond. Hun inspanningen en dergelijke. En zo deden we dat dan eigenlijk.	**L**	Geeft geen voorbeelden van revisies of iteraties.
Waren er ook revisies op de applicatie zelf?	Nee, ja, maar dat was gewoon tussendoor. Zo van: hee, let je daar nog effe op. Maar dat werd niet behandeld in zo'n bestand bijvoorbeeld op OneDrive. Dat was meer hoe ben je, doe je goed mee. Maar we gingen niet echt inhoudelijk in op de code of…		
Op welk moment gingen jullie inhoudelijk in	Zoals ik al zei, dat was echt tussen neus en oren door. Gewoon effe tussendoor.		

op de code?			
Hoe ben je met het omvangrijke probleem van project 3 omgegaan, zodat je het met code kon oplossen?	Dat is het dus. Ik kon niks met code doen, omdat…ik ben er nauwelijks mee bezig geweest. Als ik ermee bezig was, dan was het,…ja…hele kleine dingetjes. Bijvoorbeeld, een infographic op het scherm, zeg maar. Met data die je kan uitlezen. Dat had ik dus aan mijn teamgenoot gevraagd, waarmee ik het samen moest doen. Joh, hoe moet ik daaraan beginnen? Dan krijg je een paar zetjes en vervolgens vind je het wel. Je pad, zeg maar.	**L**	Geeft geen specifieke beschrijving van een probleem opdelen in deelproblemen. Leert vooral van ervaren medestudent.
Kan je een voorbeeld geven?	Meestal was het, ja, let gewoon effe op deze lijn, die regels en dan beredeneerde ik, oké, daar lag het aan. En als ik ergens niet uitkwam, dan was het ook heel makkelijk, in de zin van: hij wist het wel, komt wel goed.		
Heb je het ook wel eens op een andere manier opgelost?	Trial & error. Uitproberen, de hele tijd tot het een keer lukt.		
Op welke wijze helpt het oefenen in Grande Omega bij het oplossen van zo'n probleem?	Nou, als ik een probleem had bij projecten, dan dacht ik niet, goh, laat ik Grande Omega openen om te kijken van eh…of daar de oplossing in zit. Dat kwam niet in me op.	Ziet geen relatie tussen GO en probleemoplossen.	

Themavragen Abstraheren

vraag	**antwoord**	*nv*	*kernpunten*
Wat waren de belangrijkste onderdelen van project 3?	De belangrijkste onderdelen…de code natuurlijk. Daar draait het allemaal eigenlijk om. En dan ook wel hoe het eruit ziet uiteindelijk. De eindopdracht, zeg maar. Hoe je het gaat presenteren. Het moet er een beetje mooi uitzien. Aangezien er toch niet naar de code wordt gekeken of iets dergelijks, ga je je daarop focussen, een beetje. Het moet er gewoon leuk uitzien. Voor de gebruiker moet het gemakkelijk te gebruiken zijn. Het zorgt er ook voor dat je niet zoveel op die code gaat focussen. In de zin van: doe ik het inhoudelijk wel goed, had ik dit niet veel makkelijker kunnen doen of…daar zit je allemaal niet mee.	**M**	Kan de essentiële informatie isoleren: code, design, bruikbaarheid.
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?	Ja, zoals ik eigenlijk al zei, als ik een bepaalde functie wilde integreren in de app zeg maar, dan gingen we kijken van oké, lukt het? Mooi meegenomen. Als het niet lukt dan ja, want er wordt toch niet gekeken naar de code, hoe we dat in elkaar hebben gezet ofzo. Dus dan kunnen we focussen op hoe het eruit ziet, de Scrumdocumenten en alles. En dan komt de app er ook wel doorheen.	**M**	Prioriteert de code en noemt daarna de minder essentiële informatie. Noemt geen specifieke schematische hulpmiddelen.
Hoe maakte je het onderscheid tussen wat je heel erg belangrijk vond en wat je minder belangrijk vond? Hoe maakte je de keuzes?	Nou, de code is in die zin belangrijk, dat je iets moet hebben om te kunnen laten zien. Maar het hoeft niet ingewikkeld te zijn, als je het maar mooi kan presenteren. Dus die code is wel degelijk belangrijk, anders heb je niks. Maar buiten dat, als je het af hebt dan zorg je ervoor dat de randzaken ook allemaal in orde zijn. Dus Scrum en dergelijke. Mooi documenteren, de testen die je hebt gedaan…		
Op welke wijze helpt het oefenen	Ja, zoals ik al zei, op het moment dat je tegen problemen aanloopt dan ga je, ik tenminste, geen Grande Omega openen. Ik heb het ook	Ziet geen relatie tussen GO en het	

in Grande Omega hierbij?	niet meegemaakt in mijn omgeving dat iemand zei, joh, ga even op Grande Omega kijken hoe het kan worden. Dus ja, niet.	project.

Themavragen Algoritmisch denken

vraag	**antwoord**	*nv*	*kernpunten*
Hoe zet jij een probleem om in code?	Hoe zet jij een probleem om in code…wat bedoelen ze daar precies mee?	**L**	Beschrijft een algemene aanpak via trial & error. Gebruikt Youtube en mede-studenten als kennisbron.
Die visualisatie coderen bijvoorbeeld, hoe pak je dat aan?	In dit geval ging ik dan kijken goh, waar is hij gestopt, zeg maar. Dan vanaf dat moment ga je zelf dingen proberen. Dan ga je zelf random dingen invoeren en de ene keer, als je effe bezig bent, op een gegeven moment kom je wel tot de oplossing. Meestal pak je ook effe een filmpje erbij op de achtergrond om te kijken oké, doe ik het wel goed. En als ik er echt niet uitkwam, dan was het weer effe mijn teamgenoten vragen van joh, hoe zit dit nou ook alweer.		
Hoe ga je na of jouw code correct functioneert? (Als je merkt dat jouw code niet goed werkt, hoe ga je dan te werk?)	Nou dan ga ik 'm zelf laten uitvoeren, op mijn scherm. Als ik bijvoorbeeld een knop maak die naar een ander scherm moet gaan en hij doet het, dan doet ie het. Dus ik ga zelf testen, klopt het wel. Daarna vraag ik ook nog aan mijn teamgenoten, jongens ik heb het een en ander in elkaar gezet. Kunnen jullie er ook naar kijken. Als zij dan aangeven, het zit goed dan…zij doen dat op dezelfde manier als ik. Daar ga ik vanuit.	**L**	Controleert zelf of code werkt maar gebruikt geen testtool. Laat medestudenten code evalueren.
Hoe zorg je ervoor dat jouw code ook werkbaar is voor anderen?	Op het moment dat ik mijn onderdeel van de code pusht naar de rest, dan zet ik er een comment bij, zo van jongens, dit stukje code heb ik gemaakt voor een bepaalde functie ofzo, iets dergelijks.	**M**	Noemt één kwaliteitskenmerk.
Heb je wel eens teruggekeken naar je oude code?	Nee, niet gedaan. Misschien wel een goed idee.	**L**	Noemt geen verbeterpunt in techniek, wel in zelfeffectiviteit.
Wat doe je nu anders? Hoe uit zich dat in je code?	Uhm… ik denk meer dingen uitproberen. Ik merk dat ik daar lang mee wacht, om dingen zelf te proberen. Ja, dat eigenlijk. De code zit nog wel op hetzelfde niveau. Ik kan nu niet zeggen van eh…ik heb daar een sprong in gemaakt.		
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Eh…ja (zucht). Wat het is met Grande Omega, ik zie het meer als een tool om oefeningen steeds te herhalen en proberen te begrijpen om die toets doorheen te komen zeg maar. Plat gezegd. Dus ik heb niet zozeer als ik met Grande Omega bezig ben dat ik denk oké…ik ben niet aan het leren programmeren maar je pikt er natuurlijk wel het een en ander van. Gewoon bepaalde…dat je bezig bent met coderen zelf. Bijvoorbeeld hoe classes werken en hoe je variabelen types meegeeft. Bijvoorbeeld dat je een bepaalde variabel, ik zeg maar wat, een type interger geeft. Of een type flow, of een boolean meegeeft.		Herkent GO niet direct als een leermiddel om te coderen. Noemt wel leeropbrengsten.
Wil je nog iets kwijt over Grande Omega?	Ja, wat ik zei in het begin, dat ik bij de derde vast…loopte, omdat ik niet verder kon. Het zou fijn zijn als er een corrigeermodel zou zijn, zodat je je weet hoe je verder kan met de opgaves.		Spreekt behoefte uit voor feedbacktool.
Wat deed je als je vastliep?	Vragen aan klasgenoten. Als die het ook niet wisten dan deed ik niks. Ik dacht, het zal wel. Wachten tot de rest ook een beetje verder komt.		

Transcript interview 8 _onderzoek Grande Omega OP4 2018

Tijdstip: woensdag 20 juni 2018

Plaats: HR Rotterdam

Interviewer: AvB

Geïnterviewde: 8, studentcode 111

Algemene vragen

vraag	**antwoord**	*kernpunten*
Wat is je vooropleiding?	Bedrijfseconomie. Ik begon eigenlijk met bedrijfseconomie en ik had het niet gehaald. Dat was ook een hbo-opleiding. Vervolgens deed ik bedrijfskunde, dat is een beetje soortgelijk, dan meer managementgericht. Een daarna koos ik voor informatica.	Geen voorkennis in programmeren
Had je bedrijfskunde afgemaakt?	Nee, niet afgemaakt. Ik moest eigenlijk stoppen want ik had mijn eerstejaars vakken niet gehaald.	Geen ervaring in het ontwikkelen van programma's.
Een BSA?	Ja een BSA. En toen informatica, een totaal andere richting (lacht).	
Omschrijf je programmeer-ervaring. Wat kon je al toen je begon aan dit collegejaar?	Helemaal niks, helemaal niks. Niks. Ik had zeg maar, bij mijn twee vorige opleidingen had ik over informatiesystemen geleerd, over database en we moesten toen met een bestaande template een eigenwebsite maken en dat vond ik dan wel leuk en toen dacht ik laat ik dan maar echt daarmee iets gaan doen. Daar was mijn interesse in al geweest, werken met database, maar niet echt mogen ervaren hoe dat achter de schermen is, als het ware.	Van onvoldoende naar tussen 5 en 6.
Als je jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheid en toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?	Begin september…uhm…een onvoldoende, laat ik dat zeggen want ik wist gewoon letterlijk niets (lacht).	Vindt programmeren bij project 2 grote stap.
Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheid en na dit collegejaar, welk cijfer is dat dan? Om welke reden?	Nog steeds tussen een 5 en 6. De reden dat ik speciaal tussen een 5 en 6 geef, is, ik weet nu wel de basistermen, de basisprincipes betreffende programmeren, maar ik heb er nog steeds moeite mee om het toe te passen, daadwerkelijk voor projecten. Want daarmee zit je al te strugglen. Want project 1 was gewoon en bordspel maken, dat was knutselen dat vond ik hartstikke leuk. En project 2 was ineens, toch wel echt met programmeren. En ik had wel een grote overgang, overstap wat je kan noemen.	70% GO en 30% rest. GO heeft geholpen met DEV1 en 2
Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?	Absoluut, want wat ik zeg…van een persoon die helemaal geen kennis heeft van programmeren kan je ook niet verwachten van: schrijf vanaf scratch een code. Dan weet ik niet waar ik moet beginnen! Er zijn zoveel verschillende talen. Dus Grande Omega heeft me enorm geholpen, vooral met DEV 1 en 2. Ik merk voor Dev 3 en 4, dat ik daar nog steeds moeite mee heb. Het is niet dat het me niet helpt maar je merkt nu wel van.. dat je…een voorkennis toch nog mist. In DEV 1 en 2 werk je met Python en dan ga je overstappen ineens naar C# en dat is toch een andere, ja, programmeertaal. Dat duurt voor mij heel lang, het begrijpen van een programmeertaal. Zodoende.	Mist voorkennis bij DEV3 en 4 Studeert zelfstandig thuis maar opgaven afmaken lukt niet altijd.

Hoeveel procent is dan te danken aan het programma Grande Omega?	Eh.. zeven…70%. Ik denk dat ik sowieso naar alle hoorcolleges ga en kan volgen. Als ik een paar lessen mis, dan betekent dat dat ik ergens mee zit, andere vakken waar ik het druk mee heb. Maar ook practicumlessen, want zonder de hulp van een docent kom ik niet verder met de opdrachten. Ik probeer dan eerst thuis zelfstandig te studeren, de readers, wat erop staat, dan probeer ik wat opgaves te maken maar helaas lukt het me niet om volledig alles te kunnen invullen: een opdracht in Grande Omega.	Gaat vooral naar hoorcolleges voor hulp van docenten.

Themavragen Denken in stappen

vraag	**antwoord**	*nv*	*Kernpunten*
Beschrijf hoe je project 3 vanaf het begin aanpakte.	Eh…project 3 was heel veel struggle want ik moest ook herkansen namelijk. Het was…het idee wat we hadden van project 3 was gewoon goed maar het was meer toen het kwam op…eigenlijk code schrijven. Visualiseren met behulp van een database natuurlijk. Toen zag je al heel veel tutorials en ik wist letterlijk niet dat alleen een achtregelige code letterlijk een connectie met je database kon maken. Met behulp van die tutorials kom je daar wel achter, alleen, het was te laat. Zo heb ik m'n project niet kunnen afronden, daardoor, zeg maar.	**L**	Rudimentaire beschrijving van aanpak. Trial & error: geen methodische of systematische aanpak. Probleem wordt niet opgedeeld of zodanig geformuleerd dat het m.b.v. computer is op te lossen. Realiseert dat bij herkansing tussenstap nodig was. Start daarna een meer volgordelijke aanpak.
Weet je nog hoe dat kwam, dat het te laat was?	We waren vooral…we kwamen een beetje te laat achter dat we een database moeten opzetten dus dat we eigenlijk wat in periode 2 hadden geleerd met Analyse, dat ging ook over database, dat was de fout, of het moment dat we erachter kwamen oh, we zijn te laat. Met eerst een hele database op te zetten en vanuit de database een connectie te maken met je programmeertaal waarmee je bezig bent om überhaupt het dashboard te maken. Want dashboard komt bij bedrijfseconomie heel vaak, alleen je moest het toen in Excell maken. Excell is nogal wat makkelijker, daar hoef je niet codes voor te schrijven, dat zijn gewoon bestaande dingen. Tools waarmee je …kan maken. En hier ben je gewoon letterlijk een voor een bezig zo'n databasevisualisatie te maken…		
Wat heb je gedaan toen je merkte: we redden het niet?	Ik heb toen mijn docenten op dat moment ook al op de hoogte gebracht van: wij hebben nu wel onze database kunnen opzetten maar helaas hebben we nog niet kunnen programmeren en eh…namelijk, we wisten dat het fout was omdat we niet wisten dat je eerst die database moest opzetten om daarna te coderen. Die tussenstap, daar waren we te laat achter gekomen. En daarna hebben we vanaf het herkansingmoment, toen hebben we de draad weer opgepakt. Want we hadden de database al opgezet en vervolgens gingen we toen echt tutorials zoeken, zeg maar wat ik zeg, een connectie maken met de database. En vervolgens dat je de gegevens uit de database filtert om het in een grafiek weer te geven.		
Hoe hield je het overzicht over wat je moest doen?	Het was voor mij al duidelijk want wat ik zei, de belangrijkste gegevens waarmee we moesten werken die hadden we al, het was gewoon daadwerkelijk de laatste gedeelte, en dat is het programmeergedeelte. En met behulp van Scrum natuurlijk. Dat is ook een duidelijk overzicht van wat je bij kan houden van wat je nog moet doen en wat je nog niet moet doen.	**L**	Geeft een algemeen voorbeeld van een poging om overzicht te houden.

Hoe wist jij wat je moest doen?	Ik had toen met mijn klasgenoten afgesproken, we gaan eerst proberen, we gaan verschillende dingen proberen want je moest natuurlijk ook een menu maken want vanuit dat menu wordt je dan automatisch geleid naar het dashboard, zeg maar, de taken verdeeld. Ik was bijvoorbeeld verantwoordelijk om het menu te maken en mijn klasgenoten verantwoordelijk om alvast dingen uit te proberen wat betreft het grafiek en hoe je kan filteren met query's.		
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en waarom je dat zo deed.	Met de product owner bedoelt u? Nadat we het eigenlijk al hadden afgerond liet ik het aan onze product owner zien, ze was eigenlijk wel overtuigd want we hadden natuurlijk helemaal niks en we hadden wel wat kunnen programmeren, dat vond ze al een grote stap of ontwikkeling van ons.	**L**	Geeft geen concrete voorbeelden van revisies of iteraties, wel van feedback van de docent.
Kan je een voorbeeld geven van wijzigingen?	Kleine wijzigingen…wat ze graag terug wou zien in ons project. Bijvoorbeeld de achtergrondkleur in mijn grafiek veranderen en eventueel andere afbeeldingen gebruiken voor m'n menu.		
Hoe ben je met het omvangrijke probleem van project 3 omgegaan, zodat je het met code kon oplossen?	Hmmm…bedoelt u in de zin van toen we het in het begin niet wisten? Ja, wat ik eigenlijk daarnet ook aangaf, we gingen naar het menu kijken. Soms is het noodzakelijk om de juiste turorials te vinden, wat je eigenlijk in je programma wilt terugzien. En dat is het lastige! Want mij een keer is overkomen, toen ik het menu moest programmeren, is, heel vaak moet je de juiste term hebben om te kunnen zoeken wat je wilt programmeren voor je menu. Voorbeeld daarvan is: ik wou eigenlijk een knop hebben en als ik op die knop zou klikken, zou er een andere knop moeten verschijnen die weer me een startpagina aangeeft van de hogeschool. Bij wijze van. En dat heet bijvoorbeeld een event handler. En als ik dat niet aan een docent had gevraagd, hoe dat heet, die functie, dan was ik er niet opgekomen. Dus heel vaak is het ook…zoektermen. Dat is even belangrijk om te weten: wat wil je nou programmeren en hoe heet dat.	**L**	Geeft geen stapsgewijze oplossings-richting. Noemt een voorbeeld van leermoment.
Op welke wijze helpt het oefenen in Grande Omega bij het oplossen van zo'n probleem?	Eerlijk gezegd, Grande Omega en project komen letterlijk niet met elkaar overeen. Want bij project moet je toch gebruikmaken van allerlei andere soorten termen en technieken om überhaupt te visualiseren wat je wilt visualiseren in je project. Kijk, je leert wel de basisprincipes, van dat een code bestaat uit een string of een int en dat soort dingen. Die zijn wel gewoon hetzelfde, maakt niet uit in welke programmeertaal je bezig bent. Maar dat blijft gewoon hetzelfde. Maar echt, de daadwerkelijke codes die je bij Grande Omega leert, heb ik niet echt kunnen toepassen voor mijn project.	Ziet geen relatie tussen GO en projecten. Leert van GO de basisprincipes van coderen.	

Themavragen Abstraheren

vraag	**antwoord**	*nv*	*Kernpunten*
Wat waren de belangrijkste onder-delen van project 3?	Ja, het opzetten van database en ja, hoe wil je je data visualiseren?	**M**	Kan de essentiële informatie isoleren
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was	Nou dat is het mooie, want je krijgt de mogelijkheid om zelf te bedenken welk onderwerp je gaat kiezen. En als je eenmaal en onderwerp hebt gekozen, toevallig hadden wij een onderwerp gekozen dat het moet gaan over studenten die	**L**	Beschrijft aantal

voor de oplossing bij het project?	in Rotterdam wonen, die kiezen om hier te studeren op de locaties van de hogescholen. En naderhand als je…dat…Ik wil iets met scholen weergegeven en ik wil iets, dat betekent dat de student ook moet reizen en hoe ga je dat eigenlijk laten zien in een grafiek. Dat soort dingen moet je wel van te voren bedenken, wat je wilt. En daarna ga je zoeken naar de juiste codegegevens. Of eventueel naar de juiste filmpjes om mijn menu te kunnen maken…of grafiek. Daarna ging ik heel vaak met mijn klasgenoten overleggen met wie ik het project natuurlijk moest maken: jongens, wat denken jullie hierover, hebben jullie andere informatie gevonden of niet. En zo gingen we eigenlijk ideeën met elkaar switchen.		volgordelijke handelingen zonder schematische hulpmiddelen of niveaus.
En toen?	Toen we hadden geswitcht, gingen we ook afstemmen: ja, wat willen we nu echt laten weergegeven en is de product owner daarmee wel tevreden, of houden we haar tevreden, want daar moet je ook natuurlijk op letten. We zijn er wel vrij in, er zijn niet echt harde eisen vanuit haar, het is puur onze input geven wij aan haar door, dit kunnen wij maken. En vaak, met het gemaakte product kan zij nog iets van verbeteringen aansturen. Jongens, ik wil een andere achtergrondkleur of ik vind het lettertype niet mooi en dat soort dingen. Het is niet van eh…dat er van te voren requirements zijn vanuit de product owner. Van: jongens, ik wil dat je minimaal zes grafieken moet maken. Wij zijn er puur vrij in. Dat vind ik wel prettig aan de ene kant, vooral omdat je nog lerend bent en je weet nog niet wat je wel en wat je niet kan, dus zodoende. Ik had natuurlijk met haar een afspraak gemaakt en we gingen toen bespreken van: joh mevrouw, ik heb dit kunnen maken, vindt u dat goed, wat moet aangepast worden? Ik checkte het bij haar.		Realiseert zich dat product owner requirements kon hebben, maar stelt zelf geen specifieke requirements op.
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Wat ik zei, het is een beetje lastig omdat een onderscheid houden met projecten….want Grande Omega, daar leer je echt de basisprincipes van programmeren; wat is programmeren, en waaruit bestaat het, wat doet nou zo'n regel. Maar het helpt niet echt per direct met je project, wat ik zei, eerder.	Vindt dat GO niet bij het project helpt.	

Themavragen Algoritmisch denken

vraag	**antwoord**	*nv*	*Kernpunten*
Hoe zet jij een probleem om in code?	Haha (lacht) das echt…huh…dat is nog steeds een lastig iets …wat ik aan het ontdekken ben, hoe ik dat moet doen, want heel vaak weet ik dat gewoon niet. Wat ik zeg, YouTube helpt daar heel goed bij! (lacht)	**L**	Kan een probleem nog niet in code omzetten.
Hoe helpt YouTube daarbij?	Dat je de juiste zoektermen moet hebben om naar het juiste filmpje te komen. Om van: kan ik met die tutorial bij wijze van spreken een knopje maken die me vervolgens verwijst naar een ander scherm.		Beschrijft een persoonlijke leerstrategie via YouTube
Als je het hebt gevonden, en dan?	Heel vaak, probeer je heel snel…probeer je dat mee. En als ik dan denk: nu heeft die meneer andersoortige gegevens die hij laat zien, dan moet je het nu ook omzetten naar je eigen gegevens die je hebt.		
Hoe doe je dat?	Dat omzetten doe ik vaak van, ja…regel voor regel. Wat hij mee typt op de scherm, dat doe ik ook (lacht) en dan verander ik het naar een andere naam. Heel vaak heb je dat, wanneer je database moet opzetten en je werkt natuurlijk met query's, dat soort dingen, er is een standaardprincipe		

	voor het schrijven van een query, er zijn een paar gegevens die je moet veranderen daarin, dus…		
Hoe ga je na of jouw code correct functioneert? (Als je merkt dat jouw code niet goed werkt, hoe ga je dan te werk?)	Vaak door het te runnen. Dan zie je van oké, ik heb nu een connectie met m'n database kunnen maken (lacht). Er is gewoon zo'n knopje, dan klik je daarop en dan runt het! Daar is een programma voor (lacht). Bijvoorbeeld, wij moesten in C# coderen, en dan staat erboven zo'n knopje dat je moet drukken en dan runt het programma met een eindresultaat van wat er gebeurt als het ware. Je ziet dan wat je hebt geprogrammeerd. Wat je in code hebt getypt zie je dan als visualisatie. Het resultaat zie je.	**M**	Beschrijft twee algemene aanpakken en noemt het resultaat. Zoekt hulp bij peercoaches en klasgenoten.
Zijn er voor jou nog meer manieren om te checken of je code goed werkt?	Uhm, ja, je kan het eventueel ook navragen. Heel vaak, als er bijvoorbeeld een foutje is, waardoor het niet runt, dan kan ik het navragen aan mijn peercoach. In principe - afkloppen - is het niet echt voorgekomen dat ik hulp nodig had van mijn peercoach omdat ik nog aardig wat klasgenoten had die mij graag een handje mee wouden helpen.		
Hoe zorg je ervoor dat jouw code ook werkbaar is voor anderen?	Wat ik doe, voor mezelf, ik ga het runnen, een paar keer runnen, proberen van: werkt het nu wel of niet en is het daadwerkelijk wat ik wil hebben, het eindresultaat. Vervolgens deel je dat op GitHub. Dat is een soort Google Drive maar dan voor code, speciaal. En zodat een andere klasgenoot daar makkelijker bij kan en dat zelf ook draaiende kan houden, op z'n eigen laptop.	**L**	Noemt geen kwaliteits-kenmerken
Heb je wel eens teruggekeken naar je oude code? Wat doe je nu anders?	Nee, eigenlijk niet meer. Want ja, je moet heel veel dingen doen, dat je amper tijd hebt om terug te kijken. Wel is het nu zeg maar zo dat…ja, een verschil is het wel. We leren nu met DEV 3 en 4, leren we natuurlijk met classes te werken en toevallig had m'n klasgenootje dat uitgelegd toen ik voor mijn game moest programmeren. Dus toen zag ik wel bepaalde elementen terug inkomen, dan zeg ik wel op zo'n punt: ja, inderdaad, we wisten daarvoor niet wat classes waren, maar ik heb er wel gebruik van gemaakt en nu pas, inderdaad is het moment dat ik terugzie: oké, Grande Omega, wat die me aanbiedt om te leren, had ik toen al toegepast. Toen bestond het nog niet, want we waren natuurlijk nog niet in periode 3 en 4.	**M**	Noemt het maken van classes als leerpunt Realiseert zich dat coderen efficiënter kan. Noemt enkele aandachtspunten, maar nog geen concrete verbeterpunten.
Dus er werd al iets aangeboden…	Er werd al iets aangeboden, nu in periode 3 en 4. Maar ik had al eerder gebruik van gemaakt terwijl ik wist dat het nog niet zover was, om dat te krijgen, dat onderwerp.		
Had je al eerder classes toegepast?	Ik had eerder classes al gebruikt. Ik had een klasgenoot die mij daarover ging vertellen, en heel vaak…en we kregen een workshop van onze peer-coaches en die gingen daar ook wat over vertellen. Maar niet zo diepgaand als dat je in de betreffende periode bent, want in de betreffende periode leer je juist nu heel veel dingen over classes.		
Betekent dat, dat je nu met de classes programmeert?	Nee. Dat was eenmalig toen voor de game. Voor project 2. Maar voor project 3 hebben wij dat niet toegepast.		
Is je code nu anders dan in het begin van je studie?	Ongeveer hetzelfde. Wat ik zeg, wat ik bij mij mis, is bepaalde voorkennis over programmeertalen zelf. En ik denk, als ik dat had, dan ga je ook op een andere manier coderen want je weet, er zijn heel veel manieren van coderen die hetzelfde dingen doet. Wat ik dan denk, in mijn gedachten, misschien heb ik nu de langste code gekozen maar…terwijl het ook nog efficiënter en korter kan. Dus eh…ik denk, dat heeft echt te maken met ervaring. Naarmate		Weet dat ervaring opdoen belangrijk is.

	je heel veel gaat programmeren, naarmate je gaat oefenen, dan ga je merken: oh, acht regels code die ik heb geschreven, kan ook in twee regels. Dat is voor mij nu nog niet zover.		
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Vooral met mijn tentamens. Natuurlijk moet je je tentamen gewoon in Grande Omega maken. Ik vind het persoonlijk, voor een persoon die niet kan programmeren, ik vind het wel gewoon duidelijk. Alleen vind ik het wel jammer dat je…ja, je wordt natuurlijk gedwongen, vind ik zelf, als je geen ervaring hebt, dan wordt je natuurlijk gedwongen naar de practicumlessen te gaan en docenten hulp te vragen. Want ik zelf persoonlijk, ik kom niet doorheen. Alleen doorheen door de opdrachten. Tenzij het een makkelijk iets is, maar dat makkelijk iets hebben we natuurlijk in DEV1 gehad. Nu ga je steeds stapjes bouwen en steeds moeilijkere dingen krijgen, dus dan ben je toch afhankelijk van je docent.	Vindt dat practicumlessen noodzakelijk zijn om GO te begrijpen.	
Wil je nog iets kwijt over Grande Omega?	Ik zou het zeker nog aanraden aan de nieuwe studenten die volgend jaar, aankomend schooljaar gaan beginnen. Alleen zou ik wel de docenten aanraden bijvoorbeeld om eh…wat we afgelopen les hadden gedaan is een opdracht opengegooid en dat de docent regel voor regel gaat toelichten, jongens dit moeten jullie invullen en dat is de reden. Wat de reden erachter is, per regel langslopen. Want dan gaat de student eigenlijk begrijpen van: oh, die code doet dat en dat en hoe kan ik later toepassen? Dat soort dingen, kan je dan echt afvragen.	Heeft behoefte aan het regel voor regel nalopen van code om het te begrijpen en toe te kunnen passen.	

Transcript interview 9 _onderzoek Grande Omega OP4 2018

Tijdstip: woensdag 27 juni 2018, 08.30 - 08.45 uur

Plaats: HR Rotterdam

Interviewer: AvB

Geïnterviewde: 9, studentcode 108

Algemene vragen

vraag	**antwoord**	*Kernpunten*
Wat is je vooropleiding?	Hiervoor heb ik de opleiding mbo ICT-beheerder gedaan. Maar de eerste twee jaar heb ik het programma…programmeren gevolgd daar, dus de ICT-opleiding programmeren. Maar door gebrek aan leraren en de slechte kwaliteit leraren ben ik naar ICT-beheerder in de laatste twee jaar gegaan. En op zich heeft me dat wel goed gedaan, omdat ik een beetje van elk vak wat kennis heb meegenomen.	Voorkennis in programmeren door vooropleiding mbo. Ervaring in het ontwikkelen van programma's.
Omschrijf je programmeer-ervaring. Wat kon je al toen je begon aan dit collegejaar?	Eh…wat ik al kon, ik kon goed overweg met PHP, MySQL, ook een beetje met Python. Ik kon vooral…ik vond het leuk om systemen te bouwen maar met object oriented programming had ik nog niet veel ervaring, C# en dergelijke. Ik had wel een goeie basis om…het begrijpen van code, dat wel.	Programmeert veel in vrije tijd.
Als je jezelf een cijfer mag geven van 1 t/m 10 voor je	Uhm…tussen de 6 en een 7. Omdat ik zelf in mijn vrije tijd ook al veel programmeer, zeg maar. Dus ik doe het	

programmeervaardigheden toen je bij de opleiding begon in september, welk cijfer is dat dan? Om welke reden?	wel veel. Ik ben er veel mee bezig en daardoor zie je sneller de verbanden in andere programma's, in de code zeg maar.	
Wat voor soort programmeerwerk doe je in je vrije tijd?	Voor mijn werk heb ik een systeem gebouwd. Ik werk als bezorger, ik breng eten bij klanten thuis. Ik werk bij een afhaalzaakje en daar heb ik een systeem gebouwd waar…als er een bestelling wordt opgenomen kunnen ze op een tablet de items aanklikken en vervolgens wordt een bonnetje uitgeprint. Dat heb ik gemaakt. Voorheen was het allemaal handmatig, moesten ze het opschrijven op papier en uitrekenen met de rekenmachine en dat heb ik dus geautomatiseerd.	
Dat gebruikt het hele bedrijf nu?	Ja, ja.	
Doe je nog andere dingen, naast dat wat je voor je werk hebt gemaakt?	Momenteel even niet, dat komt door de tentamenweek, maar normaal ben ik altijd op zoek naar een leuk site-projectje ofzo.	Van 6/7 naar 8.
Wat voor soort projectjes zijn dat?	Voor mijn stage had ik bijvoorbeeld, had ik een network-stage gedaan met Cisco-switches en dergelijke. Heb ik een inventarisatiesysteem daarvoor gebouwd die met SNMP, dat is een protocol, zeg maar, die automatisch waarden uitleest uit de switches bijvoorbeeld, welke software versie staat erop.	Programmeert graag in vrije tijd.
Als je nu jezelf een cijfer mag geven van 1 t/m 10 voor je programmeervaardigheden na dit collegejaar, welk cijfer is dat dan? Om welke reden?	Een 8. Het zit niet zozeer in kennis, maar gewoon, in brede vakgebied, zeg maar. Ik heb met meerdere talen kennis gemaakt. Dat kan ik aan mijn kennis toevoegen, zeg maar.	
Hoeveel procent van dit cijfer is volgens jou te danken aan het programma Grande Omega? Aan welke cursus of activiteiten is de rest te danken?	Ik denk 20%. En 80% gewoon van de projecten. Want puur door de tijdsdruk die je dan hebt, moet je wel presteren. Misschien is dat een goed iets, weet je wel, dan gaan de leerlingen echt iets opleveren, ze moeten wel programmeren.	20% GO en 80% projecten. Tijdsdruk is stimulerende factor.
Wat kan je zeggen over die 20% in Grande Omega?	In Grande Omega hebben we eigenlijk niet geleerd om te programmeren. We hebben meer geleerd van: hoe werkt het. Wat is de theorie achter programmeren. Het is meer eigenlijk theorie. En we hebben in de eerste drie periodes eigenlijk bijna niks gecodeerd. Dat is ook wat veel leerlingen niet leuk vonden, zeg maar.	
Wat deed je dan wel in Grande Omega?	Ja, eh…alleen opdrachten maken omdat het moest, zeg maar. Niemand was er echt fan van. Meerkeuzevragen.	GO: vooral theorie, niet zelf programmeren
Je zegt: we gingen niet programmeren en ook niet coderen?	Nee, ze hebben het wel toegevoegd in de laatste periode, maar dat is alleen in OP4 gebeurd. Verder waren het alleen keuzevragen, forward en backward assignments. Dan moet je invullen wat het programma in stappen verdeelt, zeg maar. Bij stap 1 moet je invullen wat zijn de waardes nu, stap twee weer verder. Niet echt zelf programmeren.	

Themavragen Denken in stappen

vraag	**antwoord**	*nv*	*Kernpunten*

Beschrijf hoe je project 3 vanaf het begin aanpakte.	In project drie hadden wij een project gemaakt van een datavisualisatie over alle festivals in Nederland. Die hadden we gekoppeld aan Google Maps met ook nog de database van ticketmaster, dus dan kon je gelijk alle festivals zien op de juiste plek met de prijs en dergelijke informatie. Hoe heb ik het aangepakt: in ons projectgroepje hadden we besloten om gebruik te maken van React Native, dat is een Javascript , de…platform. Ik had zelf daar geen kennis van, ik liep een beetje achter, ik liep een beetje achter de feiten aan dus je moet eerst zelf leren en mijn projectteam was eigenlijk al begonnen. Dus eigenlijk heb ik meer zelfstudie gedaan en toen ik een beetje op level was kon ik wel meedoen, maar dat was wel na een aantal weken pas.		Geeft rudimentaire beschrijving van aanpak: lijkt meer op toeval gebaseerd dan op methodische of systematische aanpak. Probleem wordt niet opgedeeld of zodanig geformuleerd dat het m.b.v. computer is op te lossen. Wel worden door samenwerking ideeën geopperd en taken verdeeld op basis van voorkennis of vrijwilligheid.
Toen ben je aangehaakt bij je groepje. Hoe ging dat toen verder?	Dat ging gewoon goed. Omdat ze zagen dat ik gewoon inzet toonde en echt wat geleerd had, accepteerden ze het ook beter. En namen ze mij op sleeptouw.		
Wat deed je precies, wat voor taken voerde je uit?	Uhm, ik deed voornamelijk de documentatie. Scrum deed ik bijwerken, dat beheerde ik allemaal. Ik stuurde het een beetje aan en ik was eigenlijk een beetje de Scrum-master die alle taken…kijken wie wat heeft gedaan, status-updates bijhouden.	**L**	
Jij hebt toch aardig wat programmeerervaring …ben je niet flink de lead gaan nemen als programmeur?	Nee, heel toevallig zat er al iemand in ons groepje die heel erg goed is, en die heeft zelf heel veel gemaakt.		
Hoe hebben jullie met z'n allen dat project aangepakt? Wat gingen jullie doen?	We gingen in het begin met elkaar zitten. En dan…wat gaan we precies maken? Vaak kwamen daar wel ideeën uit en vervolgens gingen we die ideeën weer onderbouwen met hoe gaan we…welke code gaan we gebruiken, welke libraries gaan we gebruiken. Vaak gingen we vervolgens gewoon prototypes maken, gewoon code die niet mooi is, maar wel werkt zeg maar. En als dat goed beviel, dan gingen we dat verder uitwerken.		
Hoe ging dat, dat uitwerken van die prototypes?	Vaak kregen mensen gewoon taken, van hee, kan jij dit voor ons doen. Dan werd het aangeleverd, vervolgens als een persoon dat had aangeleverd, dan werd het in de groep gegooid. En dan ging iedereen kijken of hij nog zelf kon uitbreiden of verbeteren. Dus zelf code toevoegen aan het gemaakte deel…dat geleverd werd.		
Beschrijf wat je bij je revisies of iteraties (Scrum) deed en waarom je dat zo deed.	Inhoudelijk zelf wel, maar omdat de product owner niet echt kennis had van programmeren was dat niet echt eh…het was meer van oh, leuk idee, weet je wel.	**M**	Geeft algemeen voorbeeld van een iteratie. Debugger werkt stap voor stap en student volgt deze stappen, maar geeft niet aan dat denken/werken in stappen essentieel is.
Jullie hebben zelf inhoudelijk wel revisies gedaan. Wat deden jullie dan precies?	Meestal was dat gewoon aan het einde van een sprint, van Scrum, zeg maar. Aan het einde van elke periode kwamen we even bij elkaar om te kijken: wat hebben we nu, lopen we nog op schema, moeten we harder of langzamer werken en waar moeten we meer tijd aan besteden, zeg maar.		
Als jullie merkten dat iets wat betreft het programmeren net in orde was?	Vaak probeerden we het met programmacode gewoon bug-free te maken en zodra we een fout in de code zagen probeerden we gelijk dat op te lossen		

Hoe deed je dat, bug-free maken?	Nou gewoon, als je aan het programmeren bent, je hebt iets toegevoegd, dan zie je gelijk dat er een bug inzit en dan ga je kijken waar de fout zit.		
Hoe weet je dat, hoe zoek je die fout op?	Ik ga gewoon het programma langs…je hebt een debugger, zeg maar. Als je dat runt, dan gaat ie stap voor stap de code uitvoeren. En direct zie je de output, in een console line. Vaak zie je bij een bepaalde stap, dan zie je oh, hier treedt de fout op. En dan ga je kijken, hoe kan ik dit oplossen, heb ik iets verkeerd geschreven of moet ik nog wat toevoegen.		
Hoe ben je met het omvangrijke probleem van project 3 omgegaan, zodat je het met code kon oplossen?	In het begin…ja, het probleem hadden we zelf verzonnen, zeg maar. Van een kaart met daarop alle festivals en dergelijke. In het begin hadden we ook andere ideeën. Het was meer eigenlijk, tijdens het maken…van het programmeren, kwamen we steeds met nieuwe ideeën doordat het net weer een andere richting opging. Eigenlijk was het meer: go-with-the-flow-achtig. We hadden wel een globaal beeld in gedachten maar we konden op elk moment afwijken en met iets beters dat doel bereiken, zeg maar.		Volgt geen bewuste stappen. Steeds nieuwe ideeën, een globaal beeld en go-with-the- flow-aanpak.
Je had veel ideeën. Hoe maakte je dan die keuze?	Bijvoorbeeld, dan we hadden eerst ook het idee om Instagram, het meeste daarin verwerken, bijvoorbeeld mensen die op een festival waren foto's plaatsen zodat die ook in de kaart vertoond waren, maar dat hebben we geprobeerd te maken maar achteraf kwamen we tot de realisatie dat het veel werk kostte, dus hadden we dat idee geschrapt. Puur om tijd te besparen en meer te focussen op het eindproduct.		
Wat gaf de doorslag voor het idee dat is doorgegaan?	Gewoon omdat het echt nodig was. Bijvoorbeeld, de kaart van Google Maps was echt nodig. Want we wilden eigenlijk de festivals visualiseren op de plaatsen, de locatie, zeg maar. Dat moest wel. We hadden het gebaseerd op de MoSCoW principe dus: Must have, Should have en Could have. En die andere dingen waren meer de must haves, waar we op gefocust hebben.	**L**	Prioriteert requirements. Hoog niveau bij thema AB (abstraheren).
Hoe wisten jullie wat de must haves waren?	Ja, omdat het datavisualisatie was, moest je wel iets visualiseren. En met Google Maps moesten we visualiseren, dus wat je echt nodig hebt. En de tweede was de data van Ticketmaster, dus de evenementen, en die moesten we op de kaart visualiseren. Dat waren echt de twee basiscomponenten, die echt nodig waren, anders hadden we geen app…zeg maar. Konden we niks laten zien, dus dat moest echt…		
Op welke wijze helpt het oefenen in Grande Omega bij het oplossen van zo'n probleem?	Geen. Nee, geen. Misschien een heel klein beetje, alleen met types…eigenlijk niks, want je moet programmeren in een taal die we niet hebben geleerd op Grande Omega. Want Grande Omega is gebaseerd op pseudotaal. In principe, wat je tijdens alle projecten doet, is puur zelfstudie. Grande Omega heeft daar niet heel veel toevoeging op. Vind ik dan.	Vindt dat GO weinig toevoegt.	

Themavragen Abstraheren

vraag	**antwoord**	*nv*	*Kernpunten*
Wat waren de belangrijkste onderdelen van project 3?	Toch wel de Google Maps-componenten. En de API van Google Maps. Dan kunnen developers zelf leuke	**M**	Isoleert de essentiële

	dingen toevoegen op een moment zeg maar, kleurtjes, markertjes, als je erop klikt, dan gebeurt er wat. En dat hebben wij gebruikt. Een daarna hebben we de data van Ticketmaster ingezet. Dus de informatie van festivals hebben we in Google Maps verwerkt. Bijvoorbeeld uit de Ticketmaster API kunnen wij de locatie, de Geolocatie, dus de longitude-attitude…op de kaart heb je bepaalde coördinaten, die konden wij van Ticketmaster pakken en in Google Maps zetten.		informatie voor het programmeer-deel
Hoe maakte je onderscheid tussen wat belangrijk en wat minder belangrijk was voor de oplossing bij het project?	Vaak hadden we wel dingen die echt belangrijk waren bijvoorbeeld Google Maps, dat vormde echt de basis van ons project. We hadden ook bijvoorbeeld dingen, opmaak ofzo, dat was niet echt de prioriteit…	**M**	Geeft een aanpak om relevante en minder relevante zaken te scheiden
Hoe wist je dat?	Eh…even kijken hoor…we keken gewoon, wat hebben we nodig om een MVP aan te leveren. Dat is een Minimal Valuable Product. Zodat je iets kan laten zien…Het gaat erom dat je aan elke sprint, elk einde van een sprint bij de product owner een potential shippable product kan aanleveren, zeg maar. Dus een product dat werkt en eventueel kan worden geshipt zeg maar, dus gepubliceerd. En als we gingen focussen op opmaak bijvoorbeeld, en niet op Google Maps, dan heb je iets wat niet werkt. Dus we moesten eerst zorgen dat het werkte, en als het werkte pas naar andere dingen kijken. Daar maakten we het onderscheid in Must haves en Should haves.	**H**	Noemt het prioriteren van requirements
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Dat weet ik vrijwel zeker: helemaal geen. In Grande Omega leren we alleen theorie en niet echt iets over projectmanagement en dergelijke.	Vindt GO geen toevoeging, vooral theorie	

Themavragen Algoritmisch denken

vraag	**antwoord**	*nv*	*Kernpunten*
Hoe zet jij een probleem om in code?	Ik kijk gewoon wat het probleem is en ik probeer gewoon dan eh…probeer gewoon wat. Ik ga gewoon code schrijven en vaak kom ik dan…dan los ik het in kleine stappen op, het probleem. Bijvoorbeeld, ik hak het probleem op in stukjes en probeer ik een klein deel op te lossen. Als ik dat opgelost heb, probeer ik er nog iets bij toe te voegen en dan breid ik het steeds uit.	**H**	Deelt het probleem volgordelijk op in deelproblemen (hoog niveau bij denken in stappen)
Hoe ga je na of jouw code correct functioneert? (Als je merkt dat jouw code niet goed werkt, hoe ga je dan te werk?)	Vaak test ik het gewoon als ik bezig ben met programmeren. Codeer ik iets en test ik het gelijk. En als ik merk dat iets niet goed werkt, dan blijf ik altijd zolang doorgaan tot het wel werkt. Ik ben een persoon die het gelijk wil hebben opgelost.	**M**	Beschrijft via een algemene aanpak dat coderen en code testen hand in hand gaan
Hoe test je dan?	Eh, ja gewoon, ik codeer iets…meestal heb ik een webservertje dat draait, dan push ik de code en kijk ik of mijn functionaliteit doet wat ik verwachtte. Bijvoorbeeld een optelsom, als je op oké klikt, dan laat ie de output zien als ie dat niet doet, weet ik dat hij het niet doet.		
Gebruik je een debugger daarbij of doe je dat op een andere manier?	Ja vaak wel, maar voorheen niet echt. Dan was het gewoon kijken, werkt het zoals ik wil en vaak zag ik wel: oh het werkt niet of wel. Dat deed ik meer op gevoel, zeg maar.		
Hoe zorg je ervoor dat	Vaak, als ik het voor mijzelf programmeer, let ik	**H**	Noemt

jouw code ook werkbaar is voor anderen?	helemaal niet op de opmaak of dergelijke, maar als ik weet dat iemand anders de code moet krijgen, dan spendeer ik wel extra tijd om de code wat mooier te maken, Dus alles even mooi alignen (netjes in volgorde zetten) zeg maar, of comments erboven zetten van hee; dit stukje code doet dit en dit en dit, zodat de andere programmeurs weten wat de code doet en ze het makkelijker kunnen lezen, zeg maar.		kwaliteitskenmerken
Heb je wel eens teruggekeken naar je oude code? Wat doe je nu anders?	Ja, op zich is er niet heel veel veranderd in mijn stijl van programmeren alleen dat ik iets rommeliger was, zeg maar. Dus niet aligned, geen comments. Meer structuur, ja.	**H**	Noemt verbeterpunten: meer structuur door alignen en comments en door opsplitsen.
Is er ook nog een andere manier om structuur in je eigen code te brengen?	Ja, bijvoorbeeld, je kan een programma schrijven in een document. Maar je kan ook meerdere documenten maken. Bijvoorbeeld een document met alle functies erin en een document met alles…de index-pagina waar je alle front-end development ziet. Bijvoorbeeld als je een pagina opent, zie je iets op je computer, dan heb je twee files waar deze die andere aanroept. Anders krijg je zooo lange code, maar je kan het ook opsplitsen, zeg maar.		
Op welke wijze helpt het oefenen in Grande Omega hierbij?	Het onderwerp van problemen aanpakken is wel behandeld, maar dat was meer een verhaaltje van theorie, zeg maar. De toepassing daarvan in Grande Omega was niet echt van sprake. Het idee is wel besproken maar…	Benoemt essentie van denken in stappen: probleem opdelen in deelproblemen Benoemt essentie van abstraheren: eenvoudiger maken. Vindt GO een goed idee maar vindt uitvoering niet conform verwachting. Had verwacht dat GO getest zou zijn.	
Welk idee is besproken?	Eh…problemen aanpakken…zeg maar, hoe je volgens mij dingen abstract maakt. Bijvoorbeeld, je moet een figuur tekenen, een lijn moest je tekenen, en een lijn bestaat weer uit sterretjes, zeg maar. Een sterretje moet je dan vier keer toevoegen en dan heb je een lijn. En dat is, zeg maar, het probleem. Je wilt een lijn tekenen, maar wat is nou een lijn? Een lijn bestaat uit vier sterretjes dus je moet eerst het probleem oplossen: oh, ik moet eerst een sterretje tekenen, vervolgens doe je er een sterretje bijplakken tot je vier hebt, en dan heb je een lijn. Dat is eigenlijk het probleem oppakken in kleine stukjes. Dat hebben ze wel besproken, eigenlijk.		
Je zei: abstract maken, wat bedoelde je daarmee?	Daar bedoelen ze mee: zo eenvoudig mogelijk, zeg maar. Dat valt weer terug op wat ik zei, met die lijn.		
Hoe sta jij ongeveer gemiddeld, wat cijfers betreft, in Dev en Analyse?	Momenteel heb ik alles gewoon gehaald. Ik heb nog een DEV-3 herkansing. En dat was het eigenlijk. Een cijfer? Voor mijn DEV-2 had ik een 10 gehaald, Dev 1 ook een hoog cijfer, ik weet niet meer, 8,8 volgens mij. En DEV-3 had ik een 5,5 voor theorie en die praktijk moet ik herkansen.		
Wil je nog iets kwijt over Grande Omega?	Ik denk dat het een heel goed idee is geweest, Grande Omega. Maar puur op theorie gebaseerd. We hebben gezien dat het in de praktijk niet echt goed ontvangen is door leerlingen.		
Hoe komt dat volgens jou?	Eh…ja puur omdat ze het niet getest hebben, zeg maar. Het begon als idee, we kunnen leerlingen dit aanbieden. Ik vind het zelf een heel goed idee dat ze dat hebben gemaakt hoor, maar het is niet echt goed uitgevoerd, zeg maar.		
Waar zit dat dan voornamelijk in?	Ja…we kwamen naar deze school met de verwachting om development te doen en we kregen daarvoor		

	Grande Omega. Het was eigenlijk alleen maar meerkeuzevragen…het was meer leren hoe kunnen wij Grande Omega goed beantwoorden en niet hoe kunnen wij de theorie leren. Het was meer trucjes leren om de antwoorden goed te doen. En ook elk tentamen die we gehad hebben in Grande Omega, was er wel iets mis met het systeem. We hebben vier tentamens gehad en elke keer was er wel een fout in het systeem, waardoor antwoorden kwijtraken…of dat docenten een uur te laat komen. Eh...dinsdag hadden we ook een tentamen, die werd gecanceld omdat het niet goed werkte. Je hebt een kans om je goed te presenteren aan de opleiding en dat hebben ze een beetje verpest. Steeds als er weer wat slechts gebeurt, dan stapelt zich dat op. Waardoor heel veel studenten nu heel negatief naar Grande Omega kijken, zeg maar. Niemand vond het fijn om mee te werken. Stel dat ze echt iets hadden opgeleverd dat heel goed werkte, dan denk ik dat het positief ontvangen werd. Puur omdat het niet goed werkte is het negatief ontvangen en vind iedereen het eigenlijk slecht. Maar de achterliggende gedachte van het systeem is wel goed.	Staat positief tegenover GO als concept, maar vindt GO te theoretisch. Mist praktijk in programmeren.
Wat is dat volgens jou, die achterliggende gedachte?	Gewoon, de kennis die ze ons willen overbrengen. Meer theorie van programmeren, dat wel. Maar wat het ook is, het is teveel gebaseerd op de theorie, er zit bijna geen praktijk in. Dus dat studenten daadwerkelijk programmeren daar is ook niet…van sprake.	

Bijlage 4. Analyse correlatie

Variabelen: totaal aantal oefeningen; totaal tijd besteed aan oefeningen, individuele tentamencijfers DEV.

H1: Er is positieve samenhang tussen het aantal gemaakte oefeningen, de tijd die aan de oefeningen is besteed en de prestaties, uitgedrukt in de tentamenresultaten.

Correlations

		TotAmountExercises	Exercise time minute	GradeDEV
TotAmountExercises	Pearson Correlation	1	,859**	,184*
	Sig. (2-tailed)		,000	,035
	N	132	132	132
Exercise time minute	Pearson Correlation	,859**	1	,205*
	Sig. (2-tailed)	,000		,018
	N	132	132	132
GradeDEV	Pearson Correlation	,184*	,205*	1
	Sig. (2-tailed)	,035	,018	
	N	132	132	132

**. Correlation is significant at the 0.01 level (2-tailed).
*. Correlation is significant at the 0.05 level (2-tailed).

Omdat de hypothese gericht is en er tweezijdig is getoetst, wordt de p-waarde door 2 gedeeld:
TotAmountExercises (0,000)
Exercise time minute (0,018/2=0,009)
GradeDEV(0,035/2=0,0175)

TotAmountExercises - Exercise time minute, ***r* = .859, de p-waarde is 0.00 (significant bij 0.01)**
TotAmountExercises - GradeDEV, ***r* = .184, de p-waarde is 0.017 (significant bij 0.05)**
GradeDEV- Exercise time minute, ***r* = .205, de p-waarde is 0.00 (significant bij 0.01)**

Conclusie (N=132):

a. sterk positief en significant verband (.90 -.70) tussen het aantal gemaakte oefeningen en de tijd die aan de oefeningen is besteed: een student die veel oefeningen maakt, besteedt ook veel tijd aan de oefeningen. r =.859, p = 0.00.
b. zwak positief en significant verband (.40 -.10) tussen het aantal gemaakte oefeningen en de tentamencijfers: er is samenhang tussen het aantal oefeningen en de hoogte van het tentamencijfer. Studenten die vaak oefenen, behalen hogere cijfers dan studenten die minder vaak oefenen. Het verband is significant voor het 0.05 level. r =.184, p = 0.017
c. zwak positief en significant verband (.40 -.10) tussen de tijd die aan oefeningen is besteed en de tentamencijfers: er is samenhang tussen de tijd die aan de oefeningen is besteed en de hoogte van het tentamencijfer. Studenten die veel tijd aan de oefeningen besteden, halen hogere cijfers dan studenten die weinig tijd aan de oefeningen besteden. r =.205, p = 0.00.

Hypothese H1: Er is positieve samenhang tussen het aantal gemaakte oefeningen, de tijd die aan de oefeningen is besteed en de prestaties, uitgedrukt in de tentamenresultaten, wordt aangenomen.

Bijlage 5. T-toetsen

Independent T-test totaal aantal oefeningen; verschil tussen groep 1 (AA) en groep 2 (BA)

H1: AA studenten maken meer oefeningen dan BA studenten

Group Statistics

	groups	N	Mean	Std. Deviation	Std. Error Mean
amount of excercises	Above average	104	1969,78	1273,306	124,858
	Below average	28	1075,64	617,939	116,780

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
amount of excercises	Equal variances assumed	4,450	**,037**	3,596	130	,000	894,136	248,645	402,222	1386,050
	Equal variances not assumed			**5,230**	**92,371**	**,000**	**894,136**	**170,959**	**554,615**	**1233,657**

Levene's test Sig = < 0,050. Dus de varianties zijn ongelijk.

T-test: significatiegrens: 5% (0,050)

p = 0,0. Betrouwbaarheidsinterval: laagste en hoogste waarden liggen boven 0: verschil is significant.

t(92,37) = 5,23, p = < 0,05. Hypothese 1: AA studenten maken meer oefeningen dan BA studenten wordt aangenomen.

Independent T-test totaal correct attempts; verschil tussen groep 1 (AA) en groep 2 (BA)

H1: AA studenten hebben meer correcte antwoorden bij de oefeningen dan BA studenten

Group Statistics

	groups	N	Mean	Std. Deviation	Std. Error Mean
correct answers	Above average	104	810,49	595,765	58,420
	Below average	28	349,93	221,031	41,771

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
correct answers	Equal variances assumed	7,439	**,007**	4,008	130	,000	460,562	114,924	233,199	687,925
	Equal variances not assumed			**6,413**	**117,790**	**,000**	**460,562**	**71,817**	**318,342**	**602,781**

Levene's test Sig = < 0,050. Dus de varianties zijn ongelijk

T-test: significatiegrens: 5% (0,050)

p = 0,0. Betrouwbaarheidsinterval: laagste en hoogste waarden liggen boven 0: verschil is significant.

t(117,7) = 6,41, p = < 0,05. Hypothese 1: AA studenten hebben meer correcte antwoorden bij de oefeningen dan BA studenten wordt aangenomen.

Independent T-test totaal incorrect attempts; verschil tussen groep 1 (AA) en groep 2 (BA)

H1: AA studenten maken meer fouten bij de oefeningen (incorrecte antwoorden) dan BA studenten

Group Statistics

	groups	N	Mean	Std. Deviation	Std. Error Mean
incorrect answers	Above average	104	1159,29	733,921	71,967
	Below average	28	725,71	425,622	80,435

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
incorrect answers	**Equal variances assumed**	**3,638**	**,059**	**2,988**	130	**,003**	433,574	145,089	146,533	720,615
	Equal variances not assumed			4,017	74,942	,000	433,574	107,931	218,562	648,586

Levene's test Sig = >0,050. Dus de varianties zijn gelijk.

T-test: significatiegrens: 5% (0,050)

p = 0,003. Betrouwbaarheidsinterval: laagste en hoogste waarden liggen boven 0: verschil is significant.

t(130) = 2,98, p = < 0,05. H1: AA maken meer fouten bij de oefeningen (incorrecte antwoorden) dan BA studenten wordt aangenomen.

Independent T-test totaal time in minutes; verschil tussen groep 1 (AA) en groep 2 (BA)

H1: AA studenten besteden meer tijd aan de oefeningen dan BA studenten

Group Statistics

	Groups	N	Mean	Std. Deviation	Std. Error Mean
exercise time minute	Above Average	104	5163,08	3914,193	383,818
	Below Average	28	2289,39	1822,558	344,431

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
exercise time minute	Equal variances assumed	6,507	**,012**	3,768	130	,000	2873,684	762,577	1365,016	4382,352
	Equal variances not assumed			**5,572**	**96,631**	**,000**	**2873,684**	**515,703**	**1850,108**	**3897,260**

Levene's test Sig = < 0,050. Dus de varianties zijn ongelijk.

T-test: significatiegrens: 5% (0,050)

p = 0,000. Betrouwbaarheidsinterval: laagste en hoogste waarden liggen boven 0: verschil is significant.

t(96,63) = 5,57, p = < 0,05. H1: AA studenten besteden meer tijd aan de oefeningen dan BA studenten, wordt aangenomen.

T-Test Onderwijsperiode 1

H1: AA studenten doen in OP1 meer oefensessies in Grande Omega dan BA studenten.

Group Statistics

	group	N	Mean	Std. Deviation	Std. Error Mean
totamountexc	1	105	532,53	333,289	32,526
	2	27	336,33	174,642	33,610

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
totamountexc	Equal variances assumed	5,289	,023	2,951	130	,004	196,200	66,496	64,647	327,753
	Equal variances not assumed			4,195	79,968	,000	196,200	46,771	103,122	289,278

Levene's test Sig = <0,050 dus de varianties zijn ongelijk

T-test: significatiegrens 5% (0,050)

p =0,000 Betrouwbaarheidsinterval: laagste en hoogste waarden liggen boven 0: verschil is significant.

t(79,96) = 4,195, p = < 0,05

De hypothese (voor OP1) wordt aangenomen:

H1: AA studenten doen in OP1 significant meer oefensessies in Grande Omega dan BA studenten.

T-Test Onderwijsperiode 2

H1: AA studenten doen in OP2 meer oefensessies in Grande Omega dan BA studenten.

Group Statistics

	group	N	Mean	Std. Deviation	Std. Error Mean
sessions	AA	108	268,58	346,990	33,389
	BA	24	141,50	141,228	28,828

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
sessions	Equal variances assumed	3,214	,075	1,758	130	,081	127,083	72,294	-15,942	270,109
	Equal variances not assumed			2,881	90,925	,005	127,083	44,112	39,459	214,708

Levene's test Sig = >0,050 dus de varianties zijn gelijk

T-test: significatiegrens 5% (0,050)

p =0,081 Betrouwbaarheidsinterval: laagste en hoogste waarden bevatten een 0: verschil is niet significant.

t(130) =1,758, p = > 0,05

De hypothese (voor OP2) wordt niet aangenomen: *H1: AA studenten doen in OP2 niet significant meer oefensessies in Grande Omega dan BA studenten.*

T-Test Onderwijsperiode 3

H1: AA studenten doen in OP3 meer oefensessies in Grande Omega dan BA studenten.

Group Statistics

	group	N	Mean	Std. Deviation	Std. Error Mean
sessions	AA	104	832,41	531,755	52,143
	BA	28	579,64	443,062	83,731

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
sessions	Equal variances assumed	,676	,413	2,307	130	,023	252,771	109,561	36,018	469,524
	Equal variances not assumed			2,563	50,030	,013	252,771	98,639	54,650	450,891

Levene's test Sig = >0,050 dus de varianties zijn gelijk

T-test: significatiegrens 5% (0,050)

p =0,023 Betrouwbaarheidsinterval: laagste en hoogste waarden bevatten geen 0: verschil is significant.

t(130) =2,307, p = < 0,05

De hypothese (voor OP3) wordt aangenomen:

H1: AA studenten doen in OP3 significant meer oefensessies in Grande Omega dan BA studenten.

T-Test Onderwijsperiode 4

H1: AA studenten doen in OP4 meer oefensessies in Grande Omega dan BA studenten.

Group Statistics

	group	N	Mean	Std. Deviation	Std. Error Mean
totamountexc	AA	113	295,25	315,944	29,722
	BA	19	74,26	71,071	16,305

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
totamountexc	Equal variances assumed	8,681	,004	3,027	130	,003	220,985	73,009	76,545	365,425
	Equal variances not assumed			6,519	121,235	,000	220,985	33,900	153,872	288,098

Levene's test Sig = <0,050 dus de varianties zijn ongelijk

T-test: significatiegrens 5% (0,050)

p =0,000 Betrouwbaarheidsinterval: laagste en hoogste waarden bevatten geen 0: verschil is significant.

t(121,23) = 6,519, p = < 0,05

De hypothese (voor OP4) wordt aangenomen:

H1: AA studenten doen in OP4 significant meer oefensessies in Grande Omega dan BA studenten.